Aalto University
School of Engineering

Jesse Miettinen

Virtual sensor for dynamic behavior of a large flexible rotor

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 27.04.2020
Supervisor: Assistant professor Raine Viitala
Advisor: M.Sc. (Tech) Tuomas Tiainen

Aalto University
School of Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author Jesse Miettinen		
Title of thesis Virtual sensor for dynamic behaviour of large flexible rotor		
Master programme Mechanical Engineering		**Code** Eng25
Thesis supervisor Assistant professor Raine Viitala		
Thesis advisor(s) M.Sc. (Tech) Tuomas Tiainen		
Date 27.04.2020	**Number of pages** 62	**Language** English

Abstract

Preventing excessive rotor vibration is important for reducing malfunctions and wear in rotating machinery used in many industries such as paper and energy production. Rotor vibration can be measured from bearings or from the rotor surface with various sensors. The vibration measured from the bearings is not informative enough if information of the rotor surface displacement is needed. Furthermore, measuring rotor displacement from the surface can be infeasible for economic or machine assembly related reasons.

This thesis developed a novel technique for approximating the rotor displacement from bearing vibration. The deep regression technique based on long short-term memory recurrent neural networks approximated rotor center point movement of the middle cross-section in the time domain with high accuracy. The developed technique could have industrial relevance, since it could be used to monitor vibration levels and detect excessive vibration in rotors without directly measuring the rotor surface. Additionally, this technique is considered novel since such deep regression application for rotor vibration has not been proposed previously.

Keywords Deep regression, Time domain approximation, Long short-term memory, Rotor, Vibration

Aalto-yliopisto
Insinööritieteiden
korkeakoulu

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi

Diplomityön tiivistelmä

Tekijä Jesse Miettinen

Työn nimi Virtuaalinen sensori suuren ja taipuvan roottorin dynaamiseen käyttäytymiseen

Maisteriohjelma Mechanical Engineering **Koodi** ENG25

Työn valvoja Apulaisprofessori Raine Viitala

Työn ohjaaja(t) Diplomi-insinööri Tuomas Tiainen

Päivämäärä 27.04.2020 **Sivumäärä** 62 **Kieli** Englanti

Tiivistelmä

Roottorien liiallisen värähtelyn ehkäiseminen on tärkeää vikojen ja koneen komponenttien kulumien välttämiseksi eri teollisuuden aloilla kuten paperin ja energian tuotannossa. Roottorien värähtelyä voidaan mitata laakereista tai roottorien pinnalta antureilla. Laakereita mittaamalla ei kuitenkaan voida päätellä roottorien värähtelyn amplitudia tarkasti. Roottorien värähtelyn mittaaminen pinnalta voi olla haastavaa koneissa, joissa mittajärjestelmälle on rajoitetusti tilaa. Lisäksi mittausjärjestelmien toteuttamisen kustannukset voivat olla liian korkeita tietyissä sovelluksissa.

Tässä diplomityössä kehitettiin epäsuora mittausmenetelmä roottorin värähtelylle, jonka avulla värähtelystä aiheutuvaa siirtymää pystytään approksimoimaan laakeripesistä mitatusta värähtelystä. Menetelmää kutsutaan virtuaaliseksi, koska mitattavasta suureesta saadaan ilman suoraa anturimittausta informaatiota. Kyseisellä pitkä lyhytkestomuisti -neuroverkkomalliin perustuvalla syväregressiivisellä menetelmällä onnistuttiin approksimoimaan roottorin keskimmäisen poikkileikkauksen heittoa aikatasossa hyvällä tarkkuudella. Työssä esitetty menetelmä voi soveltua hyvin esimerkiksi teollisuudessa käytettyjen vaikeasti mitattavien roottoreiden värähtelyn tarkasteluun. Lisäksi työssä kehitetty menetelmä on uudenlainen, sillä roottorin dynaamisen vasteen approksimointiin aikatasossa ei ole aikaisemmin käytetty regressiivisiä neuroverkkoja.

Avainsanat Syväregressio, Pitkä lyhytkestomuisti, Aikatasossa approksimointi, Värähtely, Roottori

Preface

The Department of Mechanical Engineering has focused on rotor research for decades. This thesis builds on this great amount of previous research conducted by many people in our department. Without the previously developed accurate measurement techniques and rotor systems, no deep learning could have happened. This thesis was part of the TwinRotor project funded by Academy of Finland.

I am very grateful for Tuomas Tiainen and Raine Viitala for trusting this project to me and for all their advice in rotor dynamics and scientific practices during this thesis. I want to thank also the whole research team led by Petri Kuosmanen & Raine Viitala. The atmosphere has been very supportive and positive. I am also grateful for Risto Viitala and the research team in Lappeenranta University of Technology who provided the datasets used in this thesis. Lastly, I want to thank Kari Hiekkanen for his contribution in designing the experiments.

Espoo 27.04.2020

Jesse Miettinen

Table of contents

Nomenclature

AI	Artificial Intelligence
BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
FFT	Fast Fourier Fransform
GRU	Gated Recurrent Unit
ICT	Information and Communication Technology
LSTM	Long Short-term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
RNN	Recurrent Neural Network

1 Introduction

1.1 Background

"Software is eating the world", was how Marc Andreessen, one of the most notable technology entrepreneurs of our time, defined the state of technology in the early 2010s [1]. In retrospect, the decade enhanced the everyday life with software. It is common knowledge that smart phones and all their applications penetrated the mass markets. Likely "Software is eating the world and becoming intelligent" would be as accurate definition for the state of technology in 2020, as was Marc Andreessen's definition almost 10 years ago. Intelligence can be comprehended as combining and deducting from information. Simultaneously information can be extracted from data, which is being collected and stored in massive amounts globally every minute. World Economic Forum predicted that the data stored globally will exceed 44 zettabytes during the year 2020 [2]. 44 zettabytes equals 1000^7 bytes. This amount of data can be a rough indicator for the intractable and growing number of information sources available. The growing amount of flowing and stored data is a result of cheaper and more powerful computer hardware. Computer memory and computing power have followed Moore's law for decades, which states that computer performance doubles every other year [3].

Increasing computer performance, growing amounts of data and decades of artificial intelligence (AI) research in the past and likely in the future all contribute to software becoming more intelligent. Although consensus for the definition of AI is nonexistent, it can be considered as any algorithm capable of something humans comprehend as intelligent. During the recent years, machine learning (ML) has been the branch of AI that has advanced the most [4]. Machine learning refers to methods enabling computers to learn a function from data instead of explicitly programming the function. Neural networks (NN) represent the best performing branch of machine learning in many application areas [5]. Furthermore, neural networks have already matured to a level with industrial relevance. For example, Google offers Natural language processing (NLP) capabilities which itself uses for understanding and answering search queries [6]. Yolo is an open source neural network (NN) model which can efficiently recognize and classify objects from video [7]. Similar ML models to Yolo powers the self-driving features in Tesla cars [8].

Despite all the advancements in machine learning and neural networks, Tesla still seems to be an exception within traditional manufacturing and mechanical industries. Although some research has focused on anomaly detection and condition monitoring of machinery [9]–[12], it seems that machine learning applications are mostly limited to information and communication technology (ICT) industry. However, many novel machine learning applications will most likely be proposed for traditional industries in the near future, since the sensor technology, computing hardware and the algorithms already exist. For example, virtual sensors approximating some dynamical property which otherwise would be hard to measure could be a useful solution in various traditional industries.

Virtual sensors approximating rotor vibration could be one of the significant neural network applications of the near future. Rotors play an important role in manufacturing and mechanical industries, since many crucial devices such as motors, turbines and production machines have a rotor in their assemblies. Excessive rotor vibration can cause wear, malfunctions and reduce the end product quality, thus information of vibration amplitude is crucial. Furthermore, the signals related to vibration of a rotor in motion are periodic in nature. With periodic signals, the phenomena related to rotor vibration often repeats in a

short amount of time. Thus, the periodic signals can be used to effectively produce examples for machine learning. Hence, this thesis develops a novel virtual sensor that can approximate rotor vibration using neural networks.

1.2 Research problem

Several techniques for measuring rotor vibration directly from the rotor surface have been developed. For example, measuring rotor surface displacement in-situ with a sliding probe was demonstrated in [13]. Furthermore, multi-probe techniques exist for the direct measurement of rotor cross-sectional center point displacement [14], [15]. Unfortunately, these probe techniques can be infeasible for in-situ measurements in operation conditions. For example, rotor vibration in paper machines can be difficult to measure in-situ, since the paper web covers parts of surfaces in all rotors, and the machine might lack space for the probes. Moreover, measuring with probes can be expensive, especially if the number of rotors in a machine is high.

Despite the significant amount of research focusing on neural network applications for rotor vibration, rotor axis displacement approximation in the time domain has received less attention. With all the possible methods to examine the dynamic behavior of rotors, a large amount of data related rotor vibration, including rotor axis displacement, can be acquired. However, little is known about what kind of functions neural networks can learn and how well they perform in rotor-vibration-related time-domain-approximation problems.

1.3 Research aim

The purpose of this thesis is to develop an approximation technique for rotor lateral response using neural networks. Rotor lateral response will be approximated in the time domain as horizontal and vertical center-point movement at the middle cross-section of the rotor. The neural network will process vibration signals from the bearings in the time domain in order to produce the approximations. These vibration signals include acceleration and radial bearing forces in horizontal and vertical directions. The resulting approximation technique includes applicable recurrent neural network model and training procedure for the lateral response.

Such a virtual sensor could overcome the challenges related to probe measurement methods, since it could work without direct measurements from rotor surface. By contrast, only bearing vibrations would need to be measured. Furthermore, bearing vibrations could be cost effectively measured, for example with MEMS accelerometers [16]. Moreover, the cost of these accelerometers is low and they can fit in small space [16].

1.4 Research scope

This thesis is confined to a rotor system with one flexible rotor as defined in the standards [17]. A flexible rotor similar to the rotors in paper machines was used in laboratory conditions in order to produce part of the data used in this thesis. A simulation model also representing a flexible rotor similar to the rotors in paper machines was used for acquiring the other part of the data used in this thesis. Since the acquired data represented individual rotors, excitations transmitted through foundations and other effects of operation conditions, such as the effect of paper web, are beyond the scope of this thesis.

This thesis is limited to approximating rotor center point movement at the center cross-section. The cross-section for approximations could be chosen arbitrarily along the center axis of the rotor. However, accurately approximating the center point movement at the

middle cross-section indicates that the technique could be applicable for an arbitrary cross-section.

This thesis focuses on developing an approximation technique to proof-of-concept stage. Hence, excessive search for optimal hyperparameters for the model or the training procedure is beyond the scope of this thesis. Furthermore, the comparison between various types of neural networks is excluded. Moreover, this thesis is not focused on developing better measurement or simulation techniques, because the measurement and the simulation model were developed in earlier research. In addition, data acquisition was unnecessary, since all raw data used in this thesis was acquired in earlier research projects.

1.5 Methods

In order to develop this solution, first two existing datasets representing rotor vibration are preprocessed to comparable format. The other dataset includes measurements from a real rotor and the other data acquired by simulating a rotor. After preprocessing, both datasets include time windows of rotor center point movement at center cross-section and acceleration data near bearings at same sampling rate. Additionally, the measurement dataset includes radial bearing force data from the bearing housings.

A recurrent neural network model and the training algorithms are developed by hyperparameter optimization. Hyperparameters of the model and training algorithms are optimized by empirically comparing the test performances of multiple trained models with randomly searched hyperparameter sets. A limited set of data is used for hyperparameter optimization in order to achieve faster convergence.

Finally, the developed model structure is trained and tested in various experiments including only the other or both datasets. The purpose for these various experiments is to evaluate the generalization of the trained models and to verify their applicability. In the experiment involving both datasets, the representational capability of simulation data will be evaluated with an experiment where a model is trained with examples from simulation dataset and tested with examples from the measurement dataset.

1.6 Contribution of the thesis

This thesis proposes a novel technique for approximating the rotor center point movement of the middle cross-section from the vibration signals acquired near bearings. A recurrent neural network is used to realize this approximator. The proposed technique includes dataset preprocessing steps, applicable model structure and training algorithms. Four different experiments using data acquired from a laboratory environment and from a simulation model are used to verify the performance of the technique.

The technique can be considered novel, since recurrent neural network models have not been used for rotor center point movement approximation in the time domain. However, the proposed technique is at the proof-of-concept stage. Steps for further development of the technique are also proposed in this thesis.

2 State-of-the-art review

In order to develop a technique to approximate rotor center point movement in time domain with neural network, this chapter presents necessary background information of dynamic behavior of rotors and of neural networks. The chapter is divided in two corresponding sections. Section 3.1 describes the phenomena related to rotor vibration including resonance

and sources for excitations which affect the center point movement of a rotor. Section 3.2 describes a recurrent neural network structure specialized for sequences and vital training techniques such as backpropagation and regularization. Furthermore, Section 3.2 presents preliminary information for understanding how neural networks learn from data.

2.1 Rotor dynamics

Typically, rotor system consists of a rotor, bearings and a foundation [18] with varying shapes and sizes. A rotor in a small DC motor might have a solid cylindrical rotor with length of 40 mm and weight approximately 30 g. Conversely, a power plant turbine might be over 10 m long and weigh over 100 000 kg [19]. In contrast, the rotor in the scope of this thesis is similar to hollow and cylindrical rotors used in paper machines, which can be over 10 meters long and weigh 150 kg to 5000 kg depending on their material and shell thickness [20].

Rotors in operating conditions typically have rotational and vibrational movement. The rigidness of a rotor affects the shape of the vibration movement. Rotors are considered rigid if their flexure caused by unbalance can be neglected within the designed operational rotating speed range [21]. The rotors in the scope of this research are not considered rigid but flexible. Rotors are considered flexible if unbalance causes flexible behavior in the designed operational rotating speed range [17].

Typically, the designed operational rotating speed range for the rotors in the scope of this thesis is approximately 450 rpm to 2000 rpm. However, running a rotor at these speeds can excite vibrations in it, even though the natural frequencies of the rotor would have been designed to be higher than the maximum operational rotating speed. The vibrations under the maximum operational rotating speed can reduce the quality of the end product for example in the paper production context [22]. In order to control these vibrations, it is important to understand the dynamic behavior of rotors. Thus, this section explains the typical vibrational behavior, typical sources for excitations and concepts related to dynamic geometry of a rotor.

2.1.1 Vibrations in rotors

Vibrations in rotors occur as three different types: lateral vibrations, torsional vibrations and axial vibrations. Lateral vibrations are directed perpendicularly to the center axis of the rotor. Torsional vibrations are twisting the rotor around its axis. Axial vibrations are directed along the rotor axis. With high speed rotors, lateral vibration is the component with the most energy. Without control of lateral vibrations, rotor system can fail due to bearing wear. [19]

If a rotor has vibrational energy and it is not under influence of external forces it tends to vibrate in a natural frequency. In other words, a natural frequency is a vibration frequency of the system when it has been affected by initial disturbance only. [19] A system can have many different natural frequencies. Critical speed is a common term used in the literature referring to a rotational speed coinciding with a natural frequency [20], [22] or to the rotational speed coinciding with the lowest natural frequency [23]. This thesis refers to a rotating speed equal to a natural frequency with critical speed.

Natural frequencies follow the eigenmodes of a rotor [22]. Eigenmodes can be considered as vibrational motion patterns that are system specific. Some of the typical eigenmodes for a rotor are presented in Figure 1. The translational modes relate to the natural frequencies of a rigid rotor and the bending modes relate to the natural frequencies of a flexible rotor [22]. The natural frequencies of the rotor in the scope of this thesis correspond to the eigenmodes

presented on the right in Figure 1. Furthermore, the lowest natural frequency of the rotor corresponds to the eigenmode presented on the top right corner in Figure 1. These mentioned eigenmodes cause lateral vibrations that are also in the scope of this thesis.

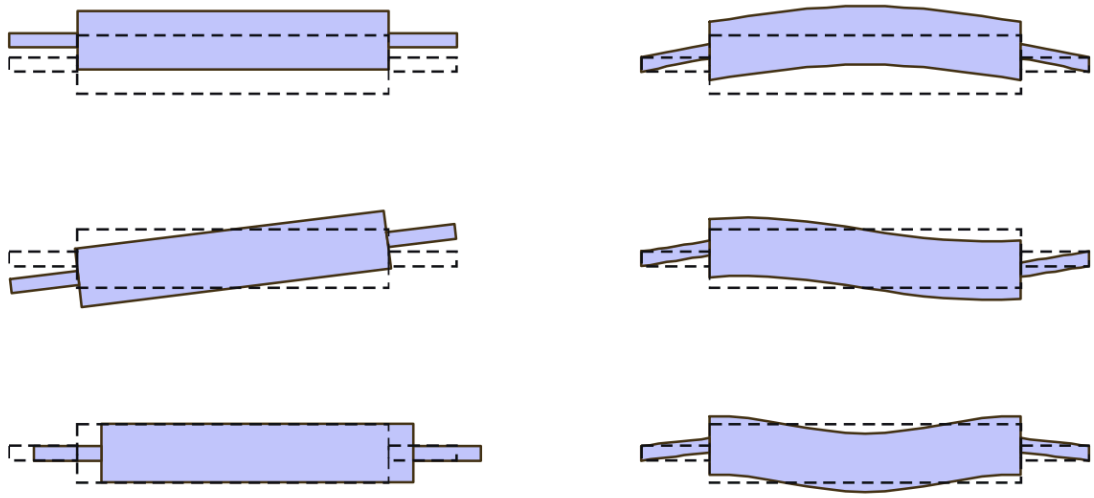


Figure 1 Some typical eigenmodes of a rigid rotor (right) and a flexible rotor (left) [24].

Resonance is a phenomenon that occurs when periodic force is applied to a system at some natural frequency of the system. The factors contributing to the amplitude of resonant vibration are demonstrated in Figure 2. The amplitude greatly increases when the ratio nears 1 between the frequencies of the applied periodic force and the natural frequency of the system [20]. The amplitude can be reduced by damping the system. Damping can be understood as friction reducing the amplitude of the vibration. Damping can be a result of the vibrating component having contact with other components of the machine.

Resonant behavior in rotors fitting in the scope of this thesis can be reduced by designing their lowest natural frequencies as high as possible or by controlling their rotating speed. For example, the natural frequencies of a paper machine roll can be approximated with Equation 1 [20].

$$f_k = \frac{k\pi}{2}\sqrt{\frac{EI}{\rho A L^4}} \qquad k = 1,2,3,\ldots \tag{1}$$

Where
- k is the number of the eigenmode
- E is the Young's modulus for the rotor material
- I is the cross sectional moment of inertia
- ρ is the thickness of the rotor material
- A is the area of cross section
- L is the distance between bearings

Typically, excessive resonance is avoided by limiting the operational speed range below the lowest critical speed [20]. However, at sub-critical speeds equal to a critical speed divided by an integer such as 1/2, 1/3 and 1/4 of critical speed, the rotor can still resonate if a periodic force is applied 2, 3 and 4 times per round [22]. These periodic forces causing resonant behavior in rotors are commonly known as excitations. Common sources for excitations are presented in the next sub-section.

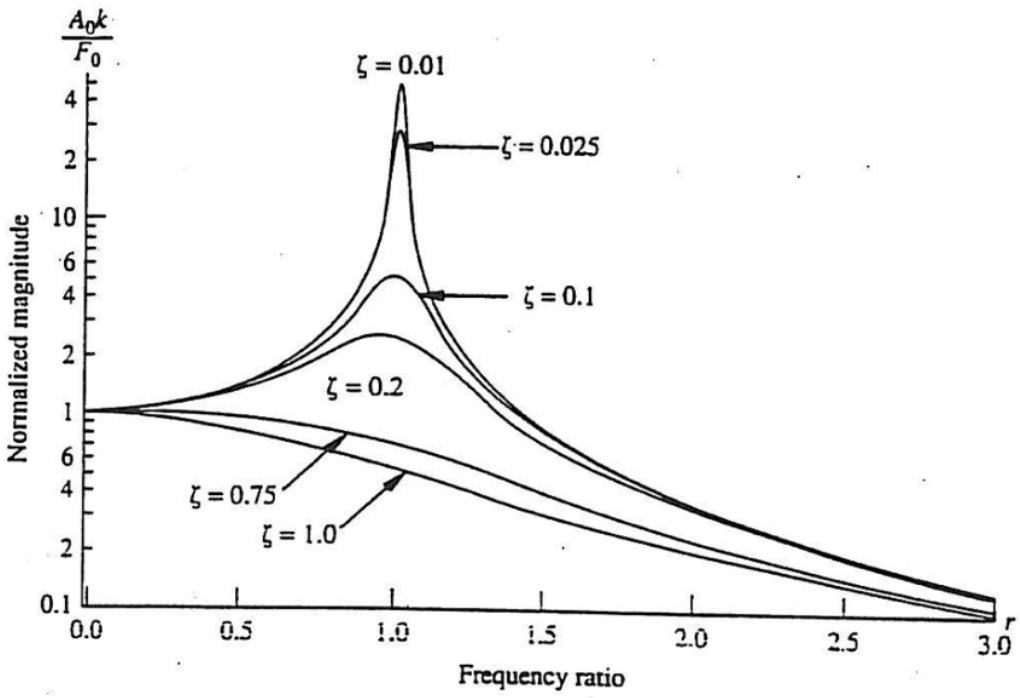


Figure 2 Amplitude of vibration depends on damping, rotors natural frequency and the frequency of excitation [20].

2.1.2 Excitations

Excitations are periodic forces vibrating rotors. Excitations can cause resonance if they occur at a natural frequency. The most common source for excitations is unbalance [20]. Other sources for excitations are asymmetries in bearings, bending stiffness variation and for example excitations transmitted through the rotor support. Asymmetries in bearings and bending stiffness variation are examples of excitations of which frequency depends on the rotational speed of the rotor. [25]

Unbalance is standardized in [26] as the "condition that exists in a rotor when vibration force or motion is imparted to it and its bearings from centrifugal forces of mass eccentricities." Mass eccentricities are standardized as "the radial distance between a center of mass and the shaft axis". Unbalance is described as the displacement of the rotational axis from the center axis of the rotor in [22], [25], [27]. This phenomenon is often divided to static and dynamic unbalance. Static unbalance is defined to be the situation when the axis of rotation is displaced parallelly from the center axis of the rotor. Dynamic unbalance is defined as the situation when the axis of rotation is not parallel to the center axis of the rotor. Static and dynamic unbalance is demonstrated in Figure 3.

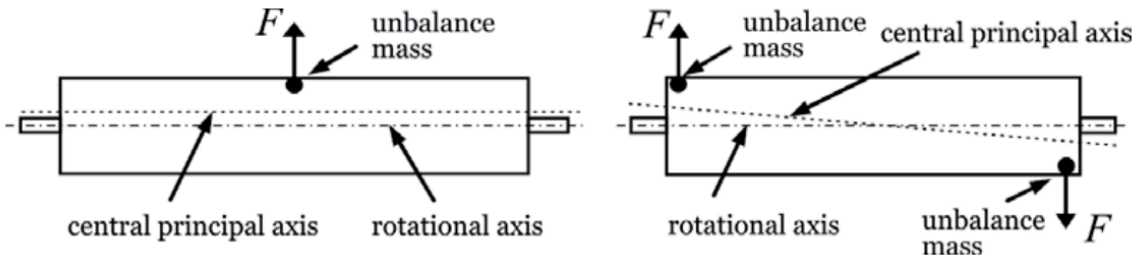


Figure 3 Static unbalance (left) and dynamic unbalance (right) of a rotor [9].

In [27] unbalance was further divided in to four categories: static, couple, quasi-static and dynamic unbalances. Static unbalance was defined as the situation where the phase and vibration amplitude at both ends of the rotor are same. Couple unbalance was defined as the situation where the vibration amplitudes at both ends of the rotor are same but the phase difference is 180 degrees. Quasi-static unbalance was defined as the situation where the phase difference is around 180 degrees but the amplitudes of vibrations at both ends of the rotor are different. Lastly, dynamic unbalance was defined as the condition when the amplitudes are different and phase angles are not entirely the same nor opposite at the ends of the rotor.

Dynamic unbalance is the most common type of unbalance in real machines. Usually it causes vibrations, rotating forces on bearings and bending which can be observed as part of dynamic runout [22]. The vibrations caused by unbalance can be observed at the same frequency as the rotational frequency [27]. Balancing is the process of changing the mass distribution of the rotor in order to reduce unbalance [22]. The unbalance of a roll after balancing is called residual balance [26]. The goal of balancing process is to force residual unbalance under some standardized tolerances.

Bending stiffness variation is caused by the difference between principal components of inertia I_x and I_y at a cross-section of a rotor. The variation causes error motion of the rotational axis when the rotor is under external load such as gravity or loads coming from the process. [23] Usually this error motion causes excitations twice per revolution. Excitations occurring twice per revolution can be caused by bending stiffness variation that was caused by oval center hole as demonstrated in Figure 4. [20] Other possible sources for bending stiffness variation are inhomogeneity of the shell material and welding seams [22]. Such excitations occurring twice per revolution can cause resonance at half-critical rotational speed.

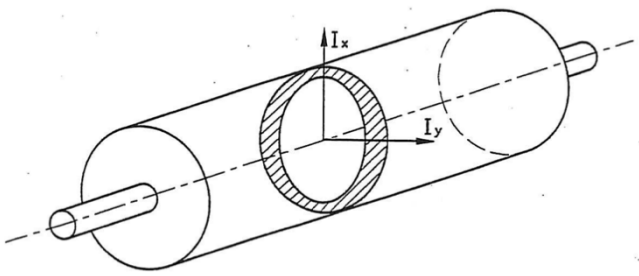


Figure 4 Difference between principal components of inertia at a cross-section caused by ovality of roll shell [15].

Bearings have also a significant excitative effect on rotors. The inner ring, outer ring and the rolling elements between bearing rings have geometrical errors, which produce excitations proportional to the rotating speed of the rotor. The main source for bearing related excitation is the roundness error of the inner ring. The roundness of inner ring is a sum of the whole assembly. As Figure 5 demonstrates, the roundness error of assembled bearing ring is affected by roundness error of the shaft and the initial roundness errors of the inner ring. Also, the adapter sleeve that might be used between the bearing and the shaft can grow the roundness error of the assembly. [28]

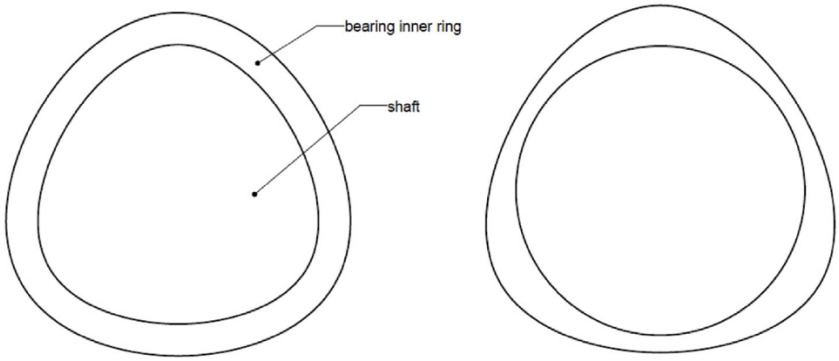


Figure 5 Triangular roundness error in bearing inner ring roundness is caused by error in roundness error of the shaft (left) and thickness variation of the inner ring (right) [23].

As mentioned, the vibrations the bearing assembly causes are interlocked with the rotational frequency of the rotor. The vibrations also depend on the type of roundness error of the bearing. For example, the triangular roundness error demonstrated in Figure 5 causes three excitations per revolution. An oval shaped assembly would cause two and quadrangular four etc. excitations per revolution. The resonant vibrations caused by these errors can be significantly reduced by manufacturing precise and fitting bearing assembly components. [25]

2.2 Neural Networks

It is important to make a distinction between neural networks (NN) and artificial intelligence (AI). Neural networks are forms of AI but represent only a fraction of its methods. Artificial intelligence is a broad concept covering everything computers are capable of and that humans could think of intelligent. Thus, the term should be used respectively. Artificial intelligence was first imagined by Alan Turing in his imitation game problem [29]. Nowadays, describing functions AI can or should perform is a trivial task, however defining artificial intelligence explicitly is a task of the opposite level of difficultness. Fortunately, a few good descriptions exist. It is defined in [30] as "a scientific endeavor that attempts to understand human cognition and an engineering discipline that tries to construct machines with human-like capabilities". Also more programming oriented views exist: "AI is any program that does something that we would think of as intelligent in humans" [31].

Neural networks, also commonly known as deep learning, represent a fraction of different machine learning (ML) algorithms, which in turn represent a bigger fraction of AI capabilities. This relation illustrated in Figure 6 can be formulated as follows: $NN \subset ML \subset AI$. In Figure 6, the term deep learning refers to neural networks. AI algorithms not classified as machine learning include methods such as answer set programming [32] and Monte Carlo tree search [33]. A common factor in all machine learning algorithms is their capability to learn structures from data without being explicitly programmed [34].

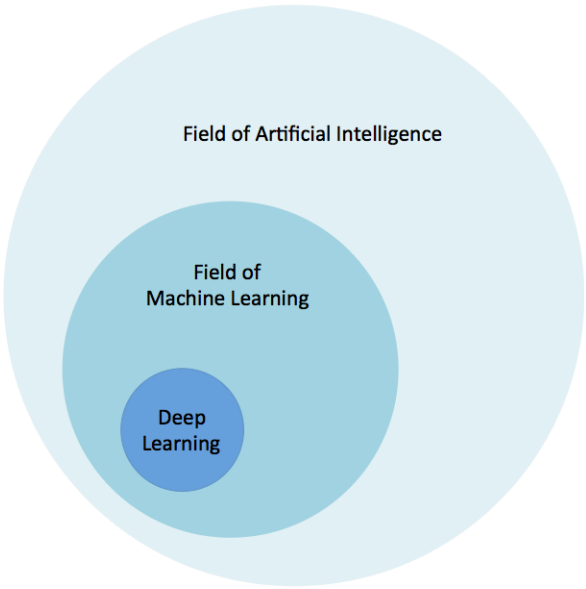


Figure 6 Relation between artificial intelligence, machine learning and deep learning (neural networks) [34].

In order to learn structures from data, machine learning algorithms often need the data preprocessed into features [35]. Usually multiple different features from the data can be used and sometimes domain knowledge is needed for engineering them. For example, a machine learning algorithm predicting the remaining lifetime of a rotor might learn effectively from a feature precomputed as follows: average rotor lifetime – (present date – date of deployment). Some domain knowledge is used for engineering such a feature. However, often feature engineering can be much more complex. According to [35], feature engineering can sometimes be the most difficult task during the whole process of developing a machine learning application.

Neural networks have an advantage over other machine learning algorithms, since they can learn from raw data [35], [36]. This means that often a considerable amount of effort in feature engineering is unnecessary. Neural networks are powerful algorithms that can learn to extract simple features from raw data and construct more complex features using the simpler features [37]. Figure 7 visualizes complex features neural network has constructed by extracting and combining simpler features from image data.

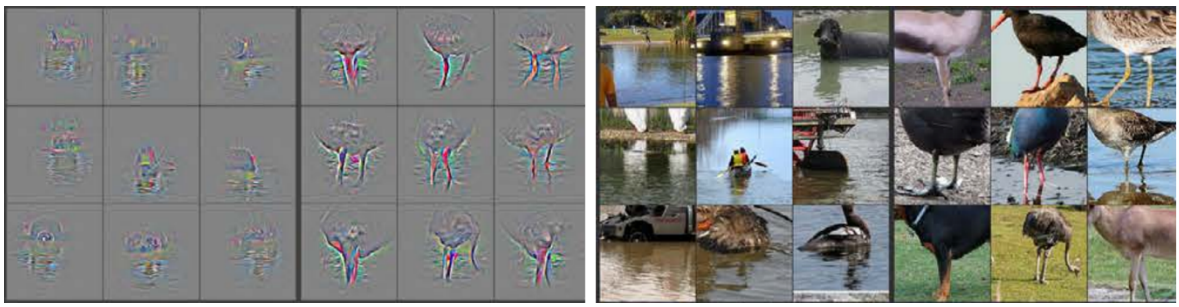


Figure 7 Visualization of complex features (left) constructed from simpler features extracted from raw image data (right) with a trained neural network [38].

Machine learning can be further divided in three different categories: supervised learning, unsupervised learning and reinforcement learning. Supervised learning approaches train the machine learning model by comparing every output of the model to predefined target values

for every input in the dataset. Unsupervised learning approaches train the machine learning model with data without explicit target values for inputs. Reinforcement learning approaches train the machine learning model by taking actions in an environment that rewards actions. [39]

Supervised learning can be further divided in classification and regression approaches. In classification approach, the targets for inputs are discrete. In regression approach the targets for inputs are continuous. [35] This thesis uses the regression approach, since the target space is a continuous space indicating the displacement of the center point in the middle cross-section of a rotor.

2.2.1 The neuron

Neural networks consist of systematically connected neurons that are capable of receiving, processing and transmitting signals. Hence, neurons are often considered analogous to biological neurons. Biological neurons are also interconnected receiving, processing and sending information via axons and synaptic links between axons [40]. This analogy between artificial and biological neurons is illustrated in Figure 8. The use of such artificial neurons for machine learning was first proposed when perceptrons were introduced [41].

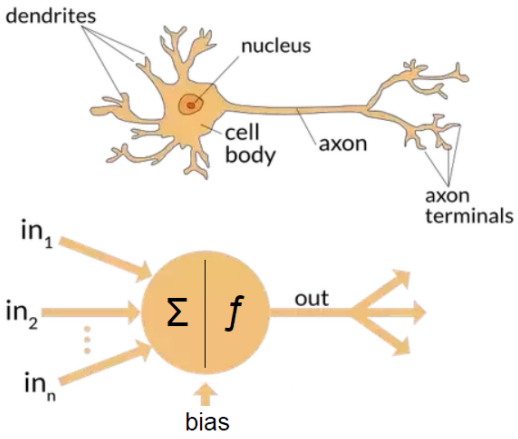


Figure 8 Analogy between an artificial and a biological neuron [42].

Figure 8 also presents typical signal processing in an artificial neuron. The neuron makes a linear combination from the input signals which is then forwarded through an activation function [35]. Equation 2 demonstrates how the neuron uses a set of weights to multiply each of the input values it receives in order to build the linear combination. These weights in every neuron of the network are optimized during the training process of the neural network. Hence, this is where the machine learning happens.

$$out = f\left(\sum_{i=1}^{n} w_i x_i + b_i\right) \tag{2}$$

Where
n is the number of inputs per neuron
i is the index of the input and corresponding weight and bias
w is the weight value used to multiply an input
x is the input value
b is the bias value
f is the activation function

Processing the linear combination through activation functions is important for solving non-linear problems. A network made of neurons without activation functions is not able to solve

the commonly known XOR-problem or problems of more non-linear complexity [37]. A number of different activation functions provide sufficient non-linear properties for the connections between neurons. However, performance of the neural network can vary with different activation functions [43]. Some of the most commonly used activation functions including most novel functions such as leaky ReLU [44] and penalized Tanh [45] are demonstrated in Figure 9.

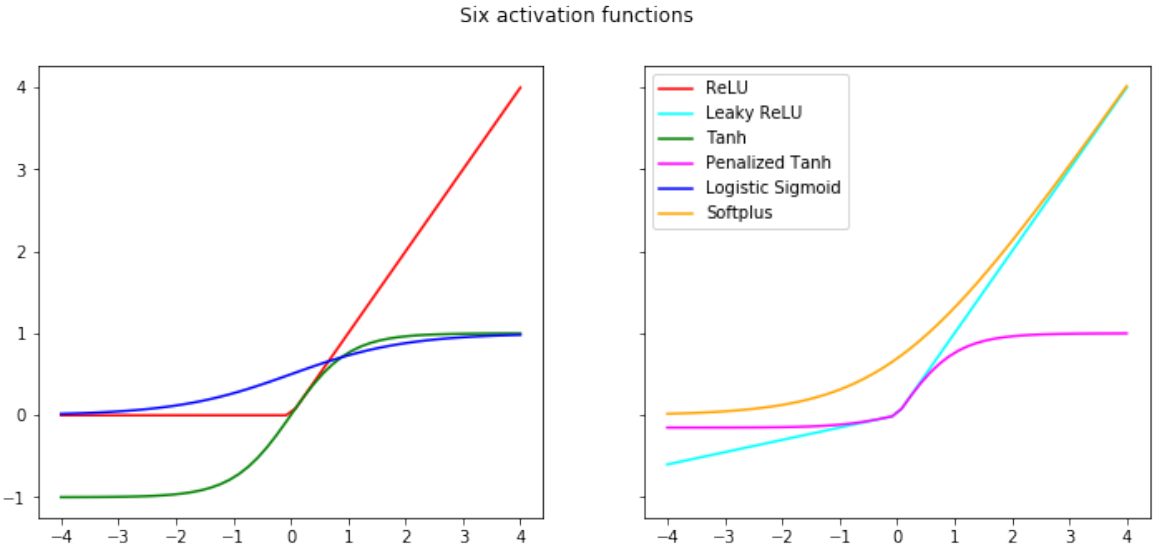


Figure 9 Some of the most commonly used activation functions.

2.2.2 Neural network structures

Neural networks consist of neurons organized in interconnected layers. Typically, the neurons in all layers are connected to the neurons in the previous and the following layer. This structure is visualized in Figure 10 showing a simple feedforward network also known as multilayer perceptron (MLP). The first layer in a feedforward layer is also called the input layer. The last layer is known as the output layer. The layers between the input and output layers are called hidden layers. [37]

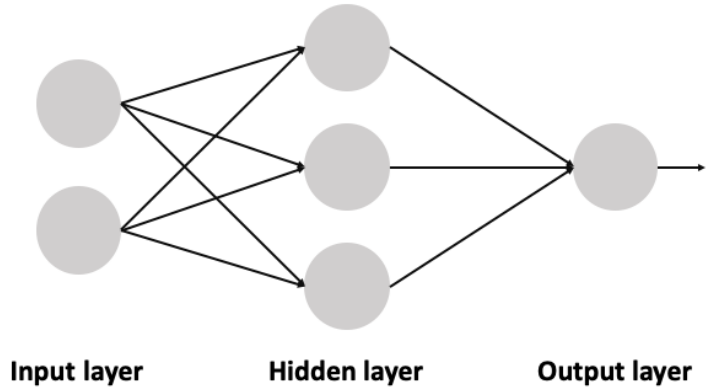


Figure 10 Multilayer perceptron with one hidden layer.

Neural network structures can vary in deepness and wideness. A network with only few hidden layers is considered as a shallow network. Correspondingly, a network with many hidden layers is considered deep [37]. Hence the use of term deep learning. Wideness relates to the number of neurons per layer. It is a common conception that increasing the number of hidden layers increases the learning capabilities of the neural network. However, it has been shown that some wideness is also needed for effective learning [46].

One of the most fundamental distinctions between different architectures is the way the neurons are connected. Figure 11 shows how neurons are connected between layers in a fully connected network such as MLP. According to [47], fully connected layers could act as universal approximators if the wideness could be increased indefinitely. However, wide fully connected layers can be unpractical since the number of connections grows the number of parameters exponentially. As Figure 11 shows, a fully connected layer with 5 neurons has 25 connections to the next layer with 5 nodes. This exponential relationship between the wideness and connections between layers can exceed available memory capacity, since for every connection a weight needs to be stored for the linear combinations in Equation 2.

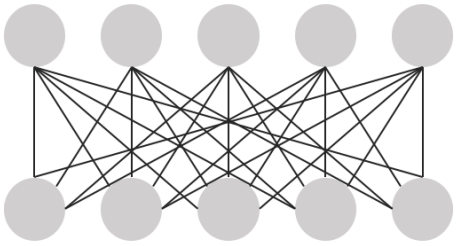


Figure 11 Fully connected layer with 25 connections.

The memory requirements for a neural network model can be reduced with parameter local connectivity and parameter sharing [48]. Local connectivity reduces the number of connections between neurons by ignoring the input connections from neurons from the previous layer with index over some predefined limit. With a predefined limit of index difference ≤ 1, the fully connected layer from Figure 11 has only 13 parameters, as is demonstrated on the left in Figure 12. With shared parameters, the number of connections between the layers on the left in Figure 12 remains the same, but memory is required for 3 parameters only. Shared parameters are indicated by shared color on the right in Figure 12.

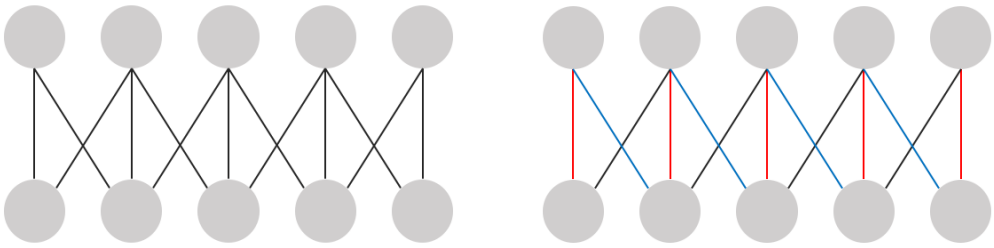


Figure 12 Number of saved parameters reduced to 13 with local connectivity limit of 1 (left) and number of saved parameters reduced to 3 with local connectivity and parameter sharing (right).

Convolutional neural networks

Convolutional neural networks (CNNs) are recognized for their accuracy in image recognition and classification problems. For example VGG-19 was state of art in the year 2014 with a classification accuracy of less than 8%. [49]. Then during the year 2015, ResNet outperformed it with an accuracy of 4.6% [50]. In 2016, YOLO-algorithm could successfully label and draw bounding boxes on objects in video with relatively high accuracy and frame rates [7].

All CNNs rely on parameter sharing and local connectivity. Typically, a convolutional neural network has a perspective field similar to w in Figure 13. CNN processes 2D data by computing dot products between the perspective field and the input data matrix, similar to z in Figure 13. The output of a CNN layer is a new data matrix, similar to h in Figure 13. Networks with this kind of convolutional layers need less resources for computation and storage than for example fully connected layers with relatively significant deepness. [37]

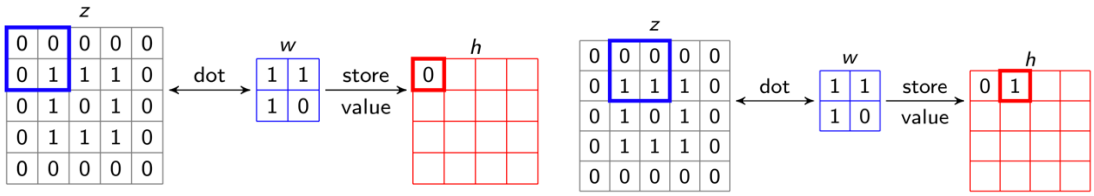


Figure 13 CNN processes 2D data with a perspective field that utilizes parameter sharing and local connectivity [40].

Recurrent neural networks

Often data might be in a sequence or sets of sequences where datapoints have an explicit order. The order in the sequences can either be dictated by time or semantics. A sequence with order dictated by semantics would be for example a sentence "Who are you?". A sequence with order dictated by time would be the signal coming from a microphone when a person says: "Who are you?" before it. These sorts of sequences can be processed with Recurrent neural networks (RNNs) [51]. RNNs can use parameter sharing between the steps in sequences, which most likely makes them often the most powerful neural network models for sequence perception.

RNNs are typically presented either as cyclic directed graphs or as unrolled directed graphs. On the left in Figure 14 the input x_t is given to a neural network f that outputs y_t at timestep t. The curved arrow is a recurrent signal connecting previous output to input in the next timestep. In the middle in Figure 14 the same recurrent neural network is unrolled over an input of five timesteps. The presented RNN could use local connectivity within one input timestep x_t and share parameters between timesteps $x_{ti}, i \in [0, \infty]$. The unrolled graph on the right in Figure 14 represents a bidirectional RNN [52]. The bidirectional RNN constitutes of two independent RNNs reading the input sequence in opposite directions.

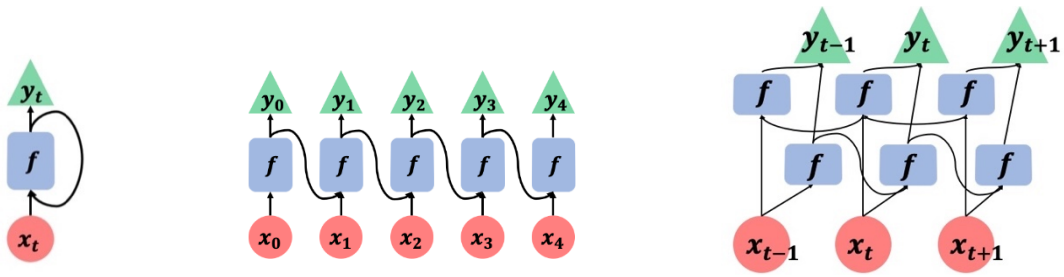


Figure 14 Cyclic graph of a RNN (left), the same RNN unrolled (mid) and a bi-directional RNN (right).

A RNN can be constructed by defining the neural network f in Figure 14 with as a perceptron similar to Figure 11. Such structure with only fully connected layers is commonly known as Vanilla RNN. Vanilla RNN and its variants were found problematic, since they struggled learning long-term patterns with the most common gradient based optimization methods

[53], [54]. During the optimization, gradients of the error term with respect to the weights of the network tend to vanish to zero or explode to infinity with Vanilla RNN.

The two most relevant state-of-the-art RNNs both solve the problem of vanishing and exploding gradients with gating mechanisms. The more commonly used model of them is the Long Short-Term Memory (LSTM) [55], [56]. The structure of LSTM is visualized with an unrolled graph in Figure 15. The other relevant state-of-the-art RNN is commonly known as gated recurrent unit (GRU) [57]. GRU has uses two gates while LSTM uses three gates, hence it is computationally less expensive model [56]. However LSTM outperforms GRU most often [56]. Thus, this thesis section is mostly concerned in LSTM.

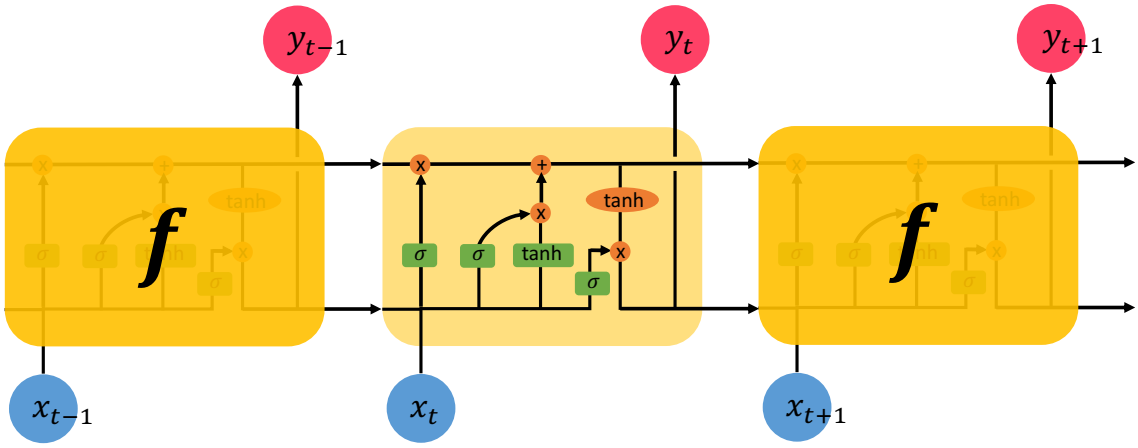


Figure 15 LSTM represented as unrolled graph with 3 timesteps.

Figure 16 - Figure 21 explain the operations LSTM computes per every timestep. The explanations follow closely [51], [58]. The notations in these figures are explained in Figure 16. The green boxes refer to a neural network layer and the markings on top of the boxes refer to the activation function used for the outputs of the layer. These boxes are also known as gates. The orange circle refers to vector multiplication or to vector addition. The rest arrows illustrate how the data flows as vectors inside the LSTM cell.

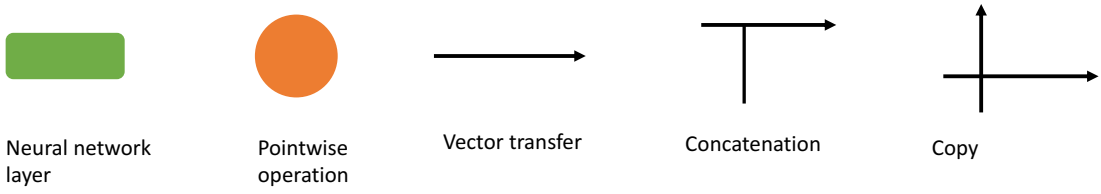


Figure 16 Operational notations inside LSTM cell.

LSTM and GRU are both constructed around a central memory state which is a vector of fixed size chosen with a priori knowledge. Central memory state or cell state C_t of a LSTM cell is shown in Figure 17. The central memory state allows LSTM to accumulate information and hold on to it over long periods of time [37]. The state is inherited from the previous recurrent step C_{t-1} and sent forward to the next recurrent step C_{t+1}.

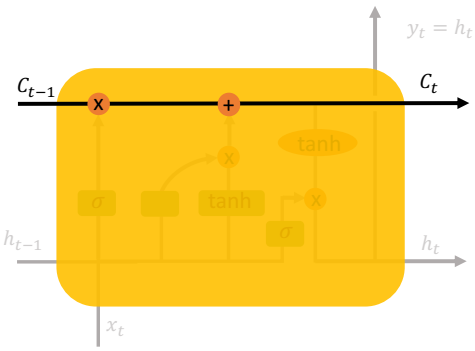
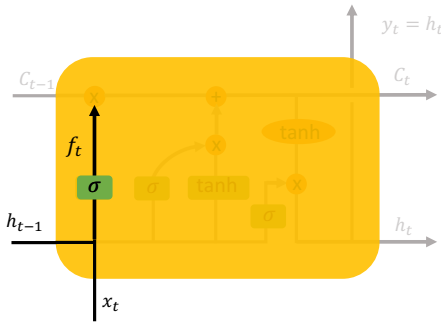


Figure 17 The state memory of LSTM cell.

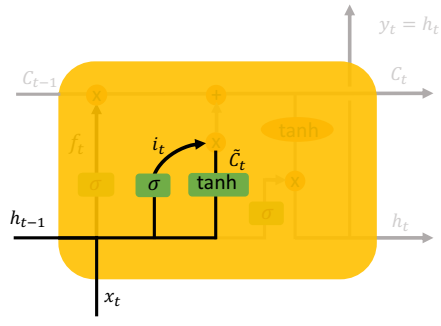
During every recurrent step, LSTM cell uses forget gate f_t first to control what to forget from the inherited central memory state from previous recurrent step. The forget gate and the corresponding neural network layer function is presented in Figure 18. The forget gate considers previous recurrent step outputs h_{t-1} and current step input x_t and makes decisions to forget values by scaling gate outputs with sigmoid function to the range 0 to 1. The values between 0 and 1 are then used to multiply values in central memory state C_t.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Figure 18 A forget gate is comprised of a linear layer with sigmoid activation function.

LSTM cell uses two neural network layers to update the central memory state. A neural network layer i_t similar to forget gate limits update values computed by the other neural network layer $\tilde{C}_t$. These operations are presented in Figure 19. The neural network layer $\tilde{C}_t$ uses tanh activation function in order to scale update values in the range between -1 to 1. The activation function for $\tilde{C}_t$ could also be any other non-linear function [37].

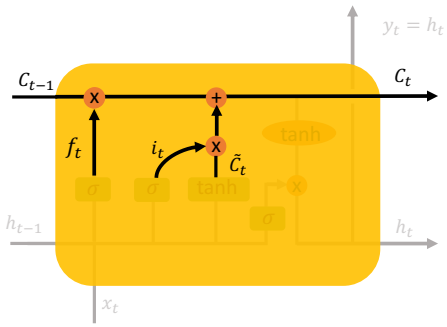


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Figure 19 An update gate is comprised of two neural network layers with tanh and sigmoid activation functions.

Memory state C_t is being updated by learned functions f_t, i_t and $\tilde{C}_t$, as show in Figure 20. The outputs of the forget gate f_t in the range 0 to 1 are multiplied with values in C_t. Forget gate decides to forget a value in C_t if the gate outputs 0. Correspondingly values higher than

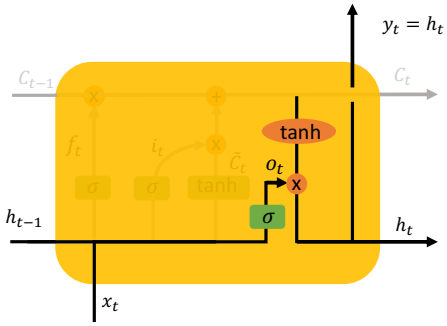
0 forces the cell to remember a fraction of the value. The outputs of the update function $\tilde{C}_t$ are controlled by another forget function i_t before their addition to the memory state C_t.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 20 LSTM cell makes updates to memory state with the outputs from forget and update gates.

Finally, the LSTM cell outputs are controlled with output gate o_t presented in Figure 21. The output gate is a similar neural network layer to the forget gate. The difference between o_t and f_t is that o_t does not control the central memory state. Instead, it controls how much of the scaled values from central memory state C_t are sent as output after the values of C_t have been scaled with tanh function.



$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

Figure 21 LSTM cell scales its output from the updated memory state with tanh function and then filters it with the output gate.

LSTM has been shown to achieve high performance in various applications. An algorithm based on LSTM learned to translate sentences between French and English [59]. Another algorithm was able to generate handwriting and Wikipedia articles in XML format [60]. Furthermore, LSTM cells were used to create state-of-the-art speech transcriber [61] and claims exist of neural network achieving 98,28 % accuracy in classifying motor faults from pure acceleration signals [62].

2.2.3 Backpropagating loss

In supervised learning the output values of the machine learning model are compared to the target values with an objective function [35]. If the task is to minimize the difference between target values and output values, then usually the objective function is called loss function, cost function or error function [37]. Typical loss functions used in supervised regression problems are Mean Squared Error (L2-loss), Mean Absolute error (L1-loss) and Huber-loss. Mean Squared Error (MSE) is probably the most used loss function for regression problems. MSE is a continuous function presented in Equation 3.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \quad (3)$$

Where N is the number of values
i is the index of value and target
y is the target value
$\hat{y}$ is the output value

Backpropagation is an algorithm calculating derivatives of the loss in terms of every parameter of the network [63]. The derivatives are calculated with the chain rule for each individual weight. In a minimization task, these derivatives are then reduced from the corresponding weights with multiple gradient descent steps [35]. Backpropagation can be considered as opposite operation to forward propagation. Forward propagation is the process of passing input signals through the neurons of a neural network in order to produce outputs. This dynamic between forward propagation and backpropagation is demonstrated in Figure 22.

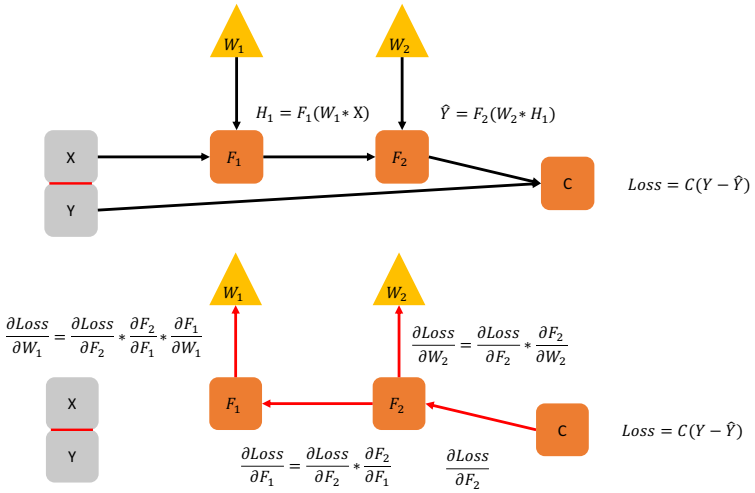


Figure 22 Forward propagation through two neurons consisting of one weight (upper) and backpropagation of the loss term through same neurons (lower).

One special form of backpropagation is backpropagation through time (BPTT). Backpropagation is considered to be BPTT when the backward passes are sent through an unrolled RNN [64]. The difference between BPTT and normal backpropagation is that in BPTT the weights are updated multiple times with gradients computed from the same loss term for every timestep in example sequence. Weights in RNNs are updated multiple times during the BPTT since RNNs share parameters among recurrent timesteps. BPTT is demonstrated in Figure 23.

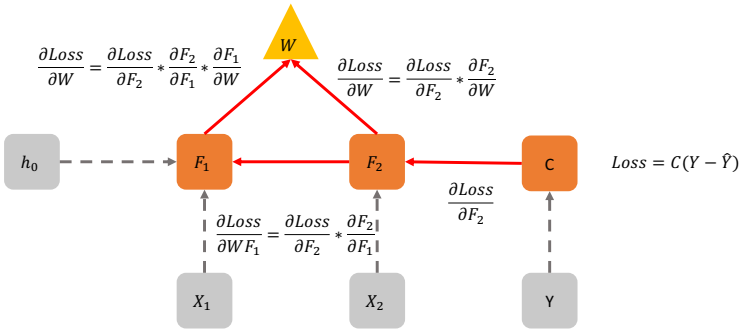


Figure 23 Recurrent neural networks are optimized with backpropagation through time.

Computing the full gradient for all possible inputs in the parameter space can be computationally infeasible with large datasets. Thus, for every gradient descent step, the gradient is approximated by sampling a batch from the dataset and then computing the derivatives of the loss term produced by the sampled data and current parameters of the network. This method of updating the weights of the network with such approximated gradients is called stochastic gradient descent (SGD) [65]. A stochastic gradient descent step is demonstrated in Equation 4.

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} C \tag{4}$$

Where $\mathbf{w}$ is the set of weights
η is the learning rate
$\nabla_{\mathbf{w}} C$ is the set of gradients

The gradient descent can be very sensitive for the learning rate η. In Figure 24 the effect of learning rate is demonstrated in a parameter space consisting of two weights. On the left, too high learning rate leads to greedy weight updates and oscillations near the optimum. On the right, too small learning rate leads to slow learning with short gradient steps. In the middle, the weights are updated with appropriate steps. Usually learning rate is chosen empirically.

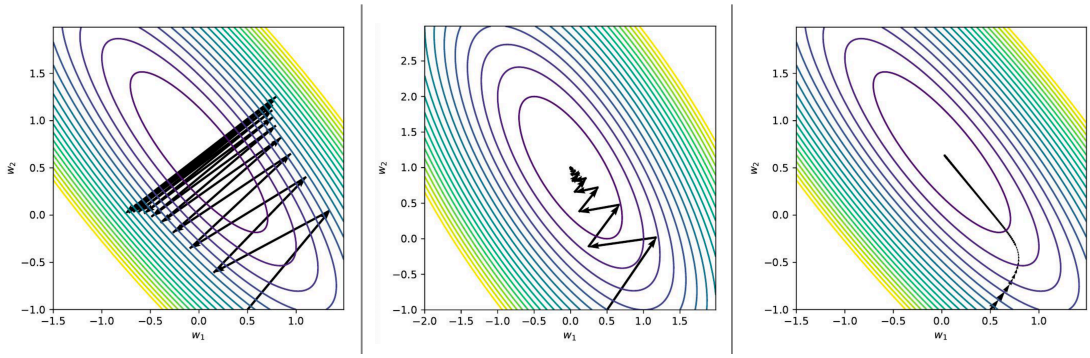


Figure 24 Effect of too high learning rate (left), fitting learning rate (middle) and too small learning rate (right) [55].

Algorithms that update the weights of ML models are often called optimizers. There are a few optimizers that rely on SGD and some mathematical improvements. An optimizer called Adam is considered often as the state-of-the-art optimizer. Furthermore, using Adam led to fastest convergence and smallest training losses in experiments to optimize some multi-layered and convolutional neural networks [66].

Adam makes choices for weight update step sizes carefully and the process is introduced in Equations 5-9 [66]. For every weight update, Adam updates momentum $\boldsymbol{m}_t$ based on the most recent backpropagation computations $\boldsymbol{g}_t$ and predefined weight parameter β_1. Momentum grows the gradient descent step sizes when the gradients point consistently to the same direction among consequent steps. Correspondingly, momentum decreases the step sizes when they point to different directions than the previous gradients. The hyperparameter β_1 controls the importance between current and previous gradients. Momentum is more thoroughly explained in [67].

After momentum, Adam computes a scaling factor $\boldsymbol{v}_t$, which depends on the squares of current and previous update steps gradients. This scaling factor is also called the second momentum [66]. Similar to first momentum, a hyperparameter β_2 controls the importance

between current and previous gradients. The scaling factor enables Adam to consistently scale gradient steps between minibatches, often resulting to consistent gradient descent.

Before the actual weight update, Adam computes initialization bias corrections $\widehat{\boldsymbol{m}}_t$ and $\widehat{\boldsymbol{v}}_t$ for the first and second momentums $\boldsymbol{m}_t$ and $\boldsymbol{v}_t$. Initialization bias corrections protect the weight updates from becoming too large, especially when small hyperparameters β_1 and β_2 are used. Finally, Adam uses these bias corrected momentums to update weights $\boldsymbol{w}_t$ with Equation 9 where η is the learning rate and ϵ is a very small value preventing divisions by zero.

$$\boldsymbol{m}_t = \beta_1 * \boldsymbol{m}_{t-1} + (1 - \beta_1) * \boldsymbol{g}_t \tag{5}$$

$$\boldsymbol{v}_t = \beta_2 * \boldsymbol{v}_{t-1} + (1 - \beta_2) * \boldsymbol{g}_t^2 \tag{6}$$

$$\widehat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1 - \beta_1^t} \tag{7}$$

$$\widehat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1 - \beta_2^t} \tag{8}$$

$$\boldsymbol{w_t} = \boldsymbol{w_{t-1}} - \eta \frac{\widehat{\boldsymbol{m}}_t}{\sqrt{\widehat{\boldsymbol{v}}_t} + \epsilon} \tag{9}$$

All the optimizers are unitless, which means that they can be used without modifications in every optimization process. Often optimizers are robust and scaling the loss function has little effect. Many optimizers can result in satisfying results and often plain SGD is enough [68]. A good way to find the best optimizer for a given problem is to simply experiment with some of them and observe the training convergence. However, when training ML models with many parameters and large datasets, Adam can likely provide robust convergence with the momentums and initial bis corrections.

2.2.4 Training principles

Neural Networks are optimized by propagating forwards and backwards batches of data from the training dataset. These small batches are called minibatches, if they include more than one example. Sometimes minibatches can also include very many examples. For example, a small minibatch is defined as anything between 32 and 512 examples while a large minibatch can include for example 10% of the total training dataset [69].

Often the training process is organized to epochs. During an epoch, every example in the training set is processed once. In order to avoid biasing the model with examples at some particular order in the training set, the training set is often shuffled between epochs. A pseudocode of a training loop is provided in Algorithm 1.

Algorithm 1 Pseudocode of a training loop

```
Do while epoch < epochs:
    Shuffle training set order.
    Do for every example in the training set:
        Pass example forward through the model.
        Backpropagate loss.
        Update weights with chosen optimizer.
    epoch += 1
```

Usually the number of epochs is set high enough to provide sufficient convergence. Convergence can be monitored with loss term development. Often there are 3 various loss terms under consideration when analyzing the performance of a neural network. The losses are training loss, validation loss and test loss. Training loss is usually compared between adjacent sampled batches of training data and/or average loss per epoch. Typically, a validation dataset is extracted from the training data, which is used frequently to measure the performance of the model during optimization. The examples in a validation dataset are not used for training, but for computing loss only. Finally, after optimization a test dataset consisting of examples also excluded from training dataset is used for unbiased evaluation of the performance of the model. Graphs similar to Figure 25 are a convenient way to monitor the performances of models in training.

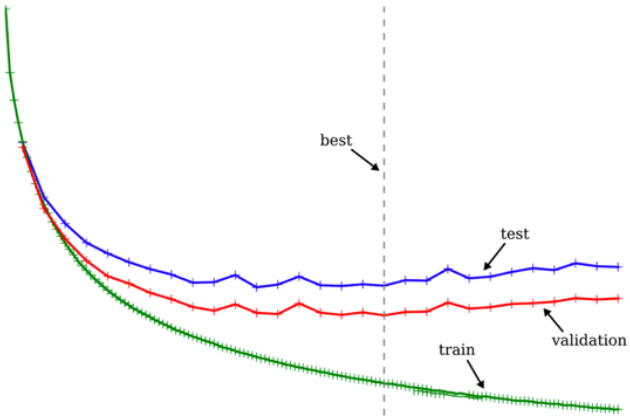


Figure 25 Typical convergence statistics and optimal time to stop training [64].

Regularization

Overfitting is a common problem in machine learning. Overfitting means that the model is over trained on the training data set, resulting in weights adjusted to the noise in the training data instead of learning "a general predicting rule" [70]. Overfit can be monitored and often precisely pointed as the stage in training when the validation performance starts to decrease while the training performance still increases. For avoiding overfitted models, there are many solutions for example early stopping, noise injection, feature scaling, batch normalization and penalty term introduction [68], [71].

Early stopping method stops the training after some number of optimization steps are not leading to decreased validation loss [68]. If the memory requirements for the weights of the model are small, an early stopping criterion does not need to be introduced to the training process. Instead the weights can be saved on multiple stages of training and then tested after all the epochs have been computed. Often however, there might not be room to save many models or training the model any longer than necessary might not be desirable. Then it might be efficient to monitor validation loss, save only a few of the latest weight sets and stop

training after the validation loss has not decreased. The stage in training for early stopping is demonstrated in Figure 25 with the line marked as “best”.

Feature scaling, also known as normalization, and batch normalization are methods used to increase consistency between the inputs. Feature scaling can be done for the whole dataset during the preprocessing stage. Feature scaling means that every feature in the dataset is scaled independently to some range that can be for example from 0 to 1. Batch normalization can have a generalizing effect on the training, by reducing covariate shift [72]. Covariate shift is caused by the updated weights that change the scale of input signals between layers during the training. Batch normalization solves this problem by normalizing the inputs $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ in a mini-batch with n inputs among each of the n dimension. A neural network can have multiple batch normalization layers.

3 Methods

This chapter presents the steps taken to develop the approximation technique for rotor center point movement at middle cross-section in the time domain. Although measurement techniques are not in the scope of this thesis, the measurement setup is presented in the beginning of this chapter, since it is crucial part of the whole proposed approximation technique.

In order to evaluate the developed technique, a set of four different experiments were designed. All four experiments were similar supervised learning experiments where rotor center point movement of middle cross-section was approximated from bearing vibration signals, as shown in Figure 26. The distinctive feature between these experiments was the contents in the training and testing datasets described in Table 1.

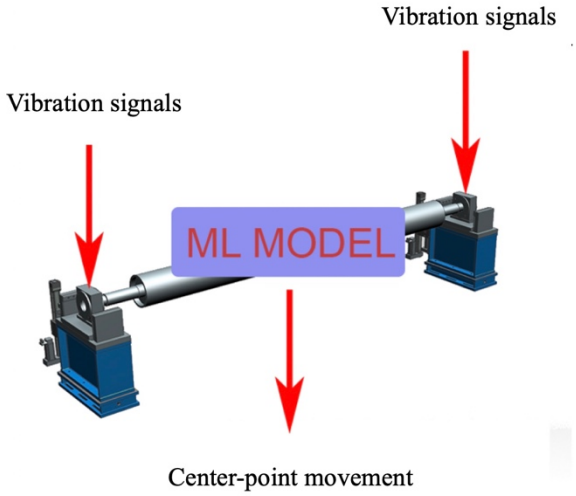


Figure 26 The experiments included supervised learning where the neural network (ML model) approximated center-point movement of the middle cross-section from vibration signals near bearing housings.

In first two experiments the neural network was trained and tested with data acquired with sensors in laboratory setting. Hence, they are referred to as observation to observation (O—>O) experiments. In first O—>O experiment, the division between training and testing data was done by randomly selecting and excluding test samples from the training data. Thus, the test dataset is sharing same physical parameter ranges with training data. Hence this experiment is referred as the O—>O interpolation experiment. The second O—>O experiment divides the observation dataset in to training set consisting of lower rotating speeds and to a test set consisting of higher rotating speeds. By forcing the trained neural network to extrapolate its approximations in a higher speed range, its performance can be evaluated more critically.

In the latter two experiments the neural network was trained with data acquired from simulation of a large flexible rotor. Only the test dataset was different between these two experiments. In the first experiment the test set was similarly randomly selected and excluded from training set and hence it is referred to as simulation to simulation (S—>S) experiment. The last experiment is referred to as simulation to observation (S—>O) experiment. In S—>O experiment, the model trained with simulation data needed to extrapolate its approximations to the observation dataset. This sort of extrapolation again enables more critical approach to evaluating performance. Furthermore, the extrapolation

from simulation domain to observation domain can also be considered as substitutive for and more useful than extrapolation in the simulation domain.

Table 1 Overview on the four experiments. Detailed descriptions of observation and simulation data is included in Table 2.

Experiment	Training data	Testing data
O→O interpolation	66% observation data from 10 to 18 Hz rotating speed	33% observation data from 10 to 18 Hz rotating speed (Data excluded from training set)
O→O extrapolation	All observation data from 10 to 12,95 Hz rotating speed	All observation data from 13 to 18 Hz rotating speed
S→S	Randomly chosen simulation data from 4 to 18 Hz rotating speed	Randomly chosen simulation data from 4 to 18 Hz rotating speed (Data excluded from training set)
S→O	Randomly chosen simulation data from 4 to 18 Hz rotating speed	All observation data from 10 to 18 Hz rotating speed

In order to measure the representational capability of the mathematical simulation of a rotor-bearing system, a neural network needs to be trained with data generated by such system and then tested with data from a physical version of such system. Conveniently the results of S—>O experiment answer to this research question. A simulation is likely representing real flexible rotor at satisfying accuracy if the model trained with the simulation data achieves high approximation accuracy in the observation domain.

This chapter presents the measurement methods for data acquisition, the used datasets, the neural network model and related training algorithms. A strong emphasis is placed upon the measurement setup and signal processing. Contents of the datasets are discussed briefly. Neural network model is presented as it is. Dataset preprocessing steps including the division to testing and training datasets are presented more thoroughly. Also, the training algorithm and the training hyperparameter optimization procedure is explained in the end of this chapter.

3.1 Observation data

The observation dataset used in this thesis was acquired with the test bench shown in Figure 27. The test bench was with only a few modifications the same as in [73] and [74]. It was built on a roll grinding machine which enabled moving the laser sensors between different cross-sections of the rotor. The foundation for the rotor was made of steel and it lay on a concrete floor. The rotor was a cylinder and its dimensions are in Figure 28. The lowest natural frequencies were known from previous research and were approximately 30.0 Hz vertically and 21.6 Hz horizontally. The rotor was attached to the foundation with spherical roller element bearings. The drive end of the rotor was at the right end of the rotor in Figure 27. The drive shaft was attached to the rotor with universal joints in order to reduce the effect of angular velocity variation. The universal joints between the drive shaft and rotor are shown in Figure ***29***.

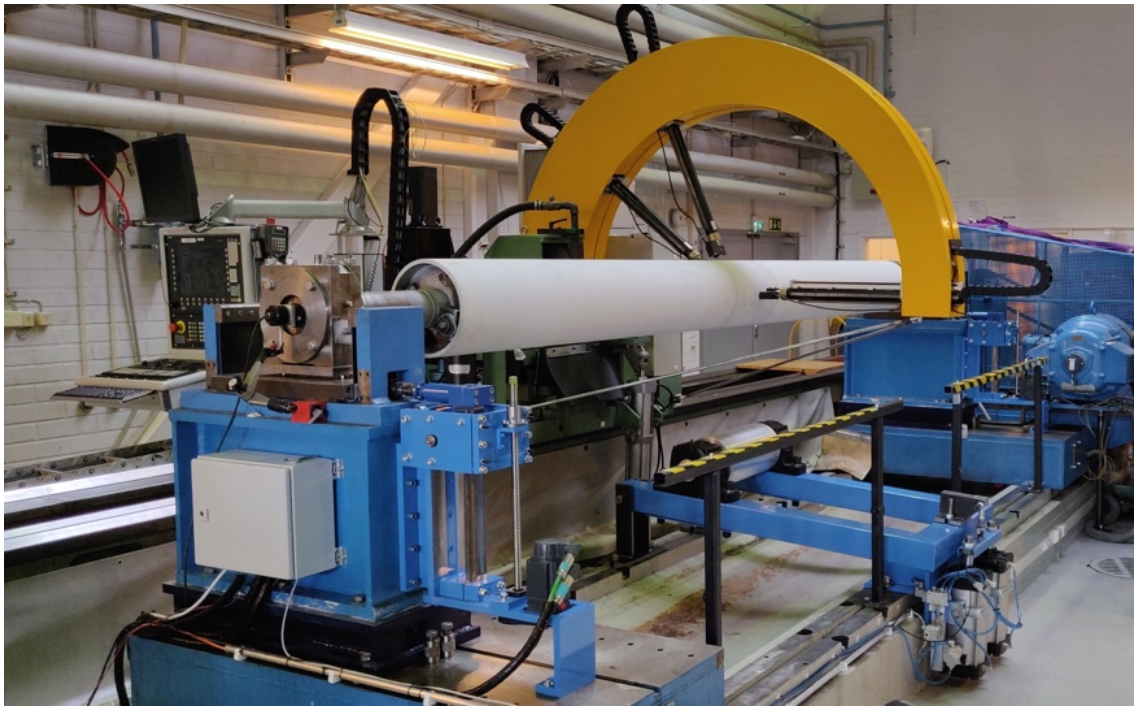


Figure 27 Laboratory test bench used for the measurements.

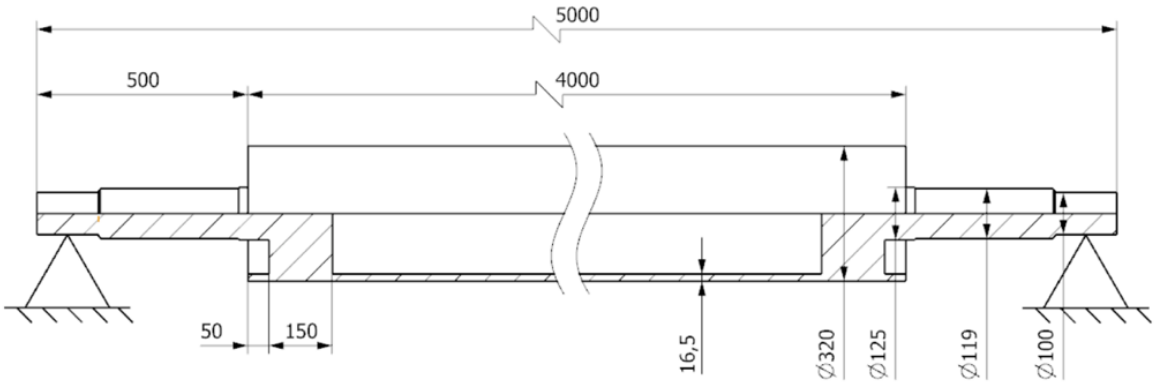


Figure 28 Dimensions of the rotor used for data acquisition [10].

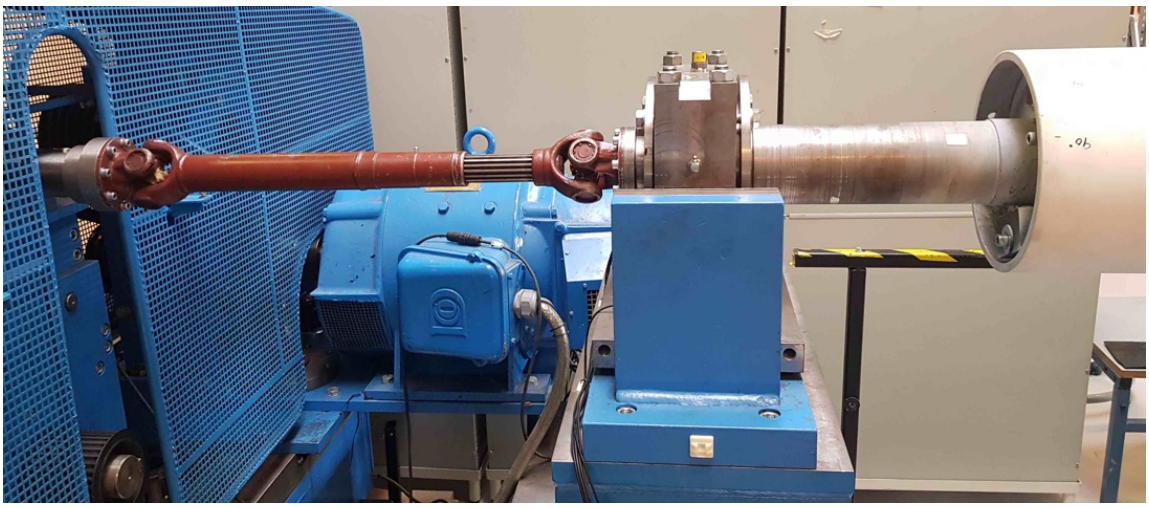


Figure 29 The drive shaft was attached to the drive-end of the rotor with universal joints.

The physical properties could be altered by controlling the rotational frequency or the support stiffness. A novel device for different horizontal support stiffnesses was developed in [75]. With the device, the horizontal stiffness could be set between values 2.04 and 18.32 Mn/m. The support stiffness controllers were assembled at the both ends of the rotor. The schematics of the controller are shown in Figure 30 and the assembled controllers can be seen in the rotor system assembly in Figure 27. The controllers lay next to both bearing housings.

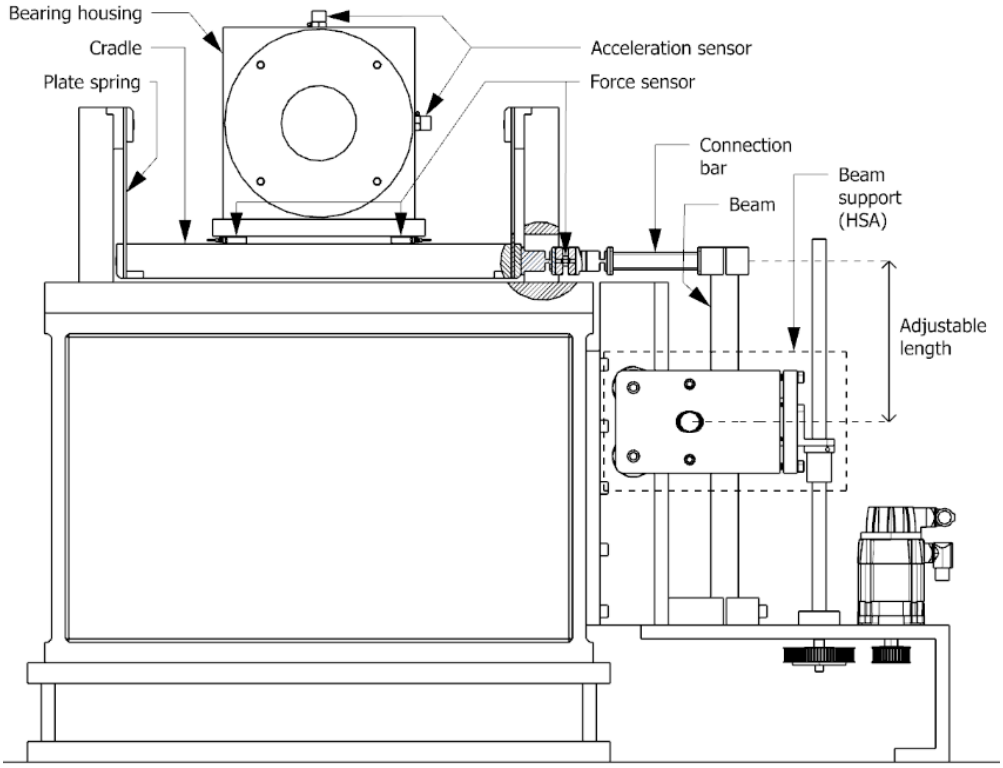


Figure 30 Assembled controller for support stiffness [75].

3.1.1 Sensors

The data was simultaneously sampled from the rotor with four laser sensors at the middle cross-section and two accelerometers and three force sensors at both rotor bearing housings. The sampling was coordinated by a rotary encoder (Heidenhein ROD 420). The encoder was used as an external sample clock which enabled 1024 samples per round to be drawn at the same angular positions. The rotary encoder was attached to the free end of the rotor as shown in Figure 31. The accelerometers measuring horizontal and vertical acceleration were placed on top and on the side of the bearing housings as shown in Figure 30 and Figure 31.

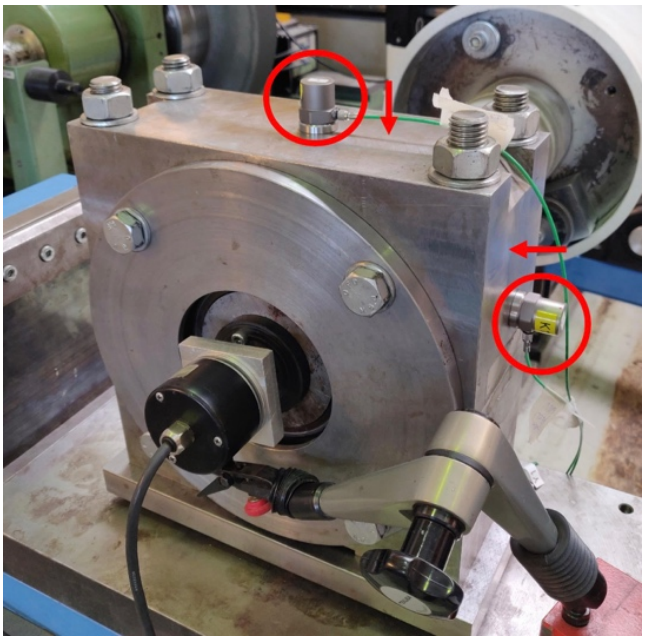


Figure 31 Accelerometers were placed on top and on the sides of the bearing housings.

The force sensors that measured radial bearing forces were attached near the bearing housings. The radial bearing force setup was first developed in [76]. The original design enabled measuring forces in the horizontal direction. Measuring this force was made possible by supporting the bearing housings with plate springs and leaving an airgap below the beds the bearing housings were rested upon. The airgap is pointed by the yellow arrow in Figure 32. The measurement technique was further developed in [73]. In the developed version, two force sensors were added under each bearing house. This enabled vertical force measurements. With this measurement setup all forces went through force sensors. Force sensor positions and respective measurement directions are pointed with red circles and arrows in Figure 32.

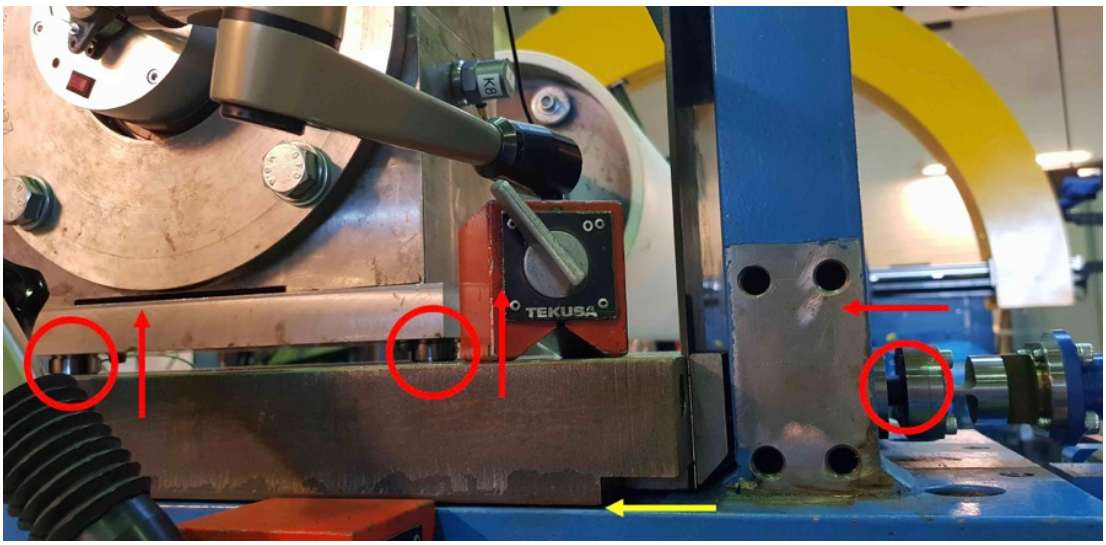


Figure 32 Horizontal and vertical force sensors (red) and airgap (yellow).

The center point movement of the rotor was measured with the four-point method [28]. With four-point method the roundness error of the rotor can be extracted from the center-point movement at any cross-section [77], [14], [78]. This center-point movement without roundness error can be considered as the dynamic lateral response of the rotor. Computing this dynamic response with four-point method requires four sensors measuring the rotor

surface displacement at different angles. The angles used for the measurements were $[0^{\circ}, 38^{\circ}, 67^{\circ}, 180^{\circ}]$. A schematic diagram for this measurement setup is shown in Figure 33 with sensor positions $[S_1, S_2, S_3, S_4]$ corresponding to the previous angles. The assembled measurement device with 4 laser sensors measuring the rotor response at the middle cross-section of the rotor is shown in Figure 27. The laser sensors used in the measurements were the same as in [22] (Matsushita Nais LM300).

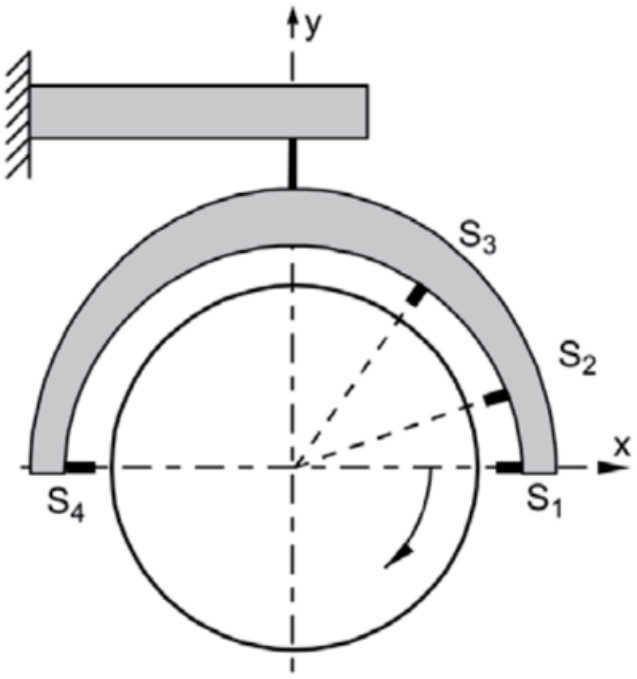


Figure 33 Schematic diagram for measuring center point movement of a cross section with four-point method [9].

The dataset was measured originally for this research [75]. Data was sampled at 31 different stiffness values between the range 18,32 and 4,25 Mn/m. In [75], the sampling procedure was described as follows:

1. The support of the beam was driven to its the stiffest point
2. The rotor was accelerated to its starting frequency 4 Hz
3. 100 revolutions were measured
4. The rotating frequency was incremented 0.05 Hz
5. 100 revolutions were measured
6. Steps 4 and 5 were repeated until the rotating frequency was at its final value limited by the safety margin (no crossing of the first mode natural frequency)
7. The rotating frequency was decreased to its starting frequency 4 Hz
8. The support of the beam was driven 10 mm downwards to decrease the horizontal stiffness
9. Steps 3 – 8 were repeated until the horizontal stiffness adjuster was at its lowest point

3.1.2 Measured signals

The center point movement of the middle cross-section was computed with the four-point method from the laser sensor signals. The four-point method computes the center-point movement horizontally from sensor signal S_1 and vertically from S_3 by extracting the roundness profile shifted to the sensor direction from raw displacement signal. The center-

point movement was computed on per-revolution basis. Examples of center-point movement computed horizontally and vertically with the four-point method are shown in Figure 34. The corresponding frequency domain computed with fast fourier transform (FFT) is shown in Figure 35.

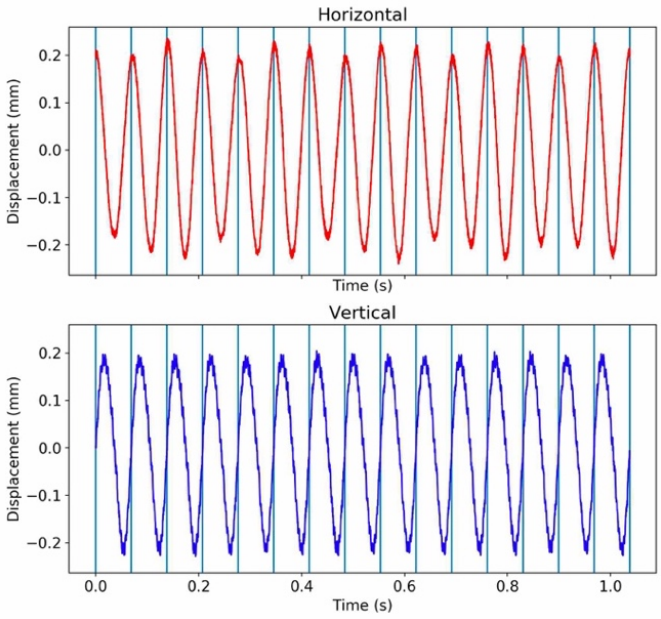


Figure 34 Example of center-point movement in time domain computed with four-point method. The blue vertical lines mark the first points of the next round.

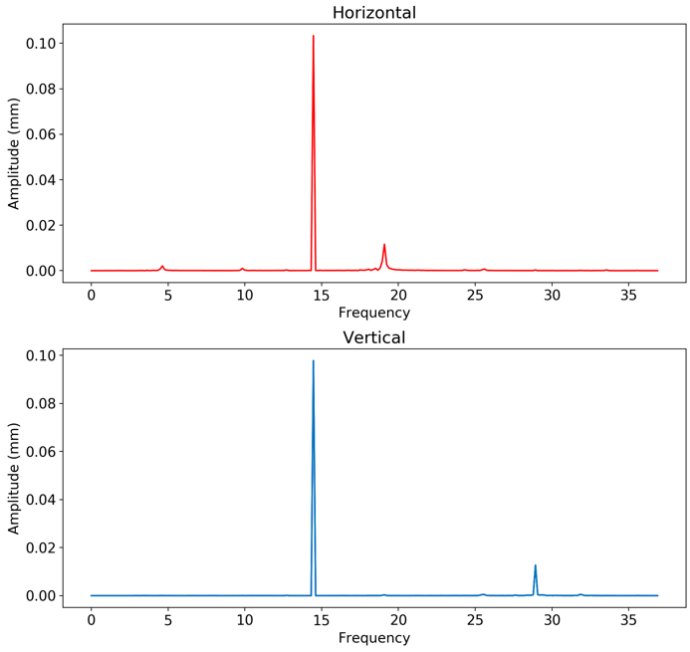


Figure 35 The horizontal and vertical frequency domain computed with fast fourier transform. The highest frequency peak corresponds to the rotating speed.

Rotor vibration is mechanically transmitted to the bearing housings. The bearing vibration was measured with force sensors and accelerometers. Examples of measured force signals are shown in Figure 36 and examples of measured acceleration signals are shown in Figure 37.

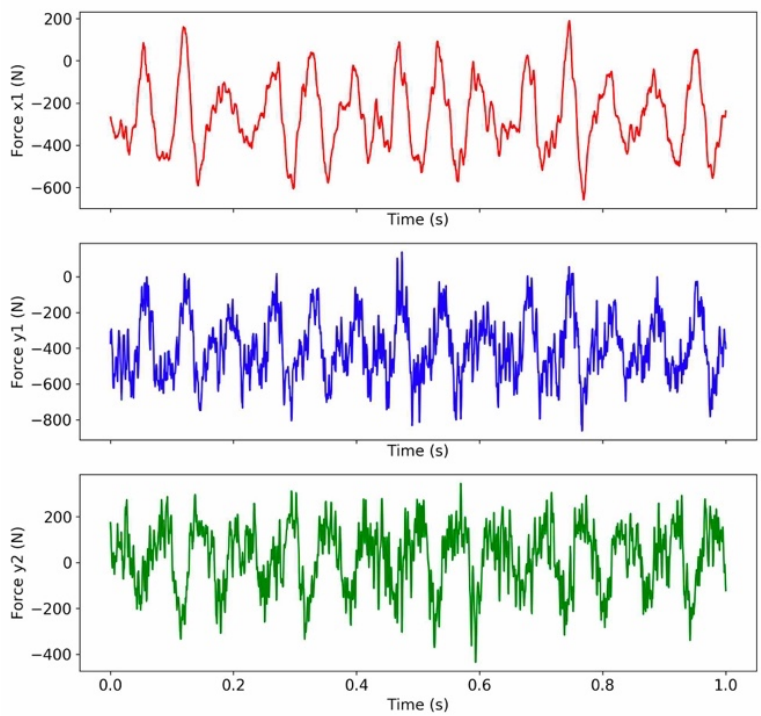


Figure 36 Force sensor signals (left) from one bearing housing, x1 (top) is the horizontal signal, y1 (middle) and y2 (lowest) are vertically directed signals.

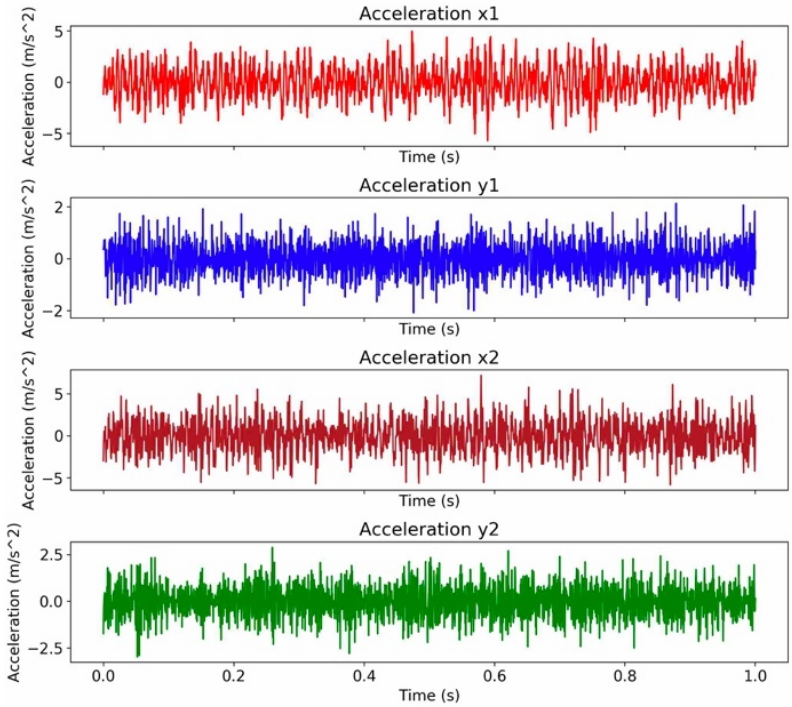


Figure 37 Accelerometer signals horizontally (x1 and x2) and vertically (y1 and y2) from both bearing housings.

All measured signals were resampled to 2000 Hz sampling frequency from initial sampling frequency which varied on the rotating speed during sampling. The differences in sampling frequencies could have caused unnecessary difficulties during training. The initial sampling frequencies for 10 Hz and 18 Hz rotating speed were for example 10240 datapoints per second and 18432 datapoints per second respectively. The resampling was done with resample -function in the signaltools provided by SciPy [79].

3.2 Simulation data

Simulation data was generated with a mathematical model of a paper machine rotor in constant motion. The data was acquired in 9 second time windows. Each time window included displacement data horizontally and vertically from three different positions in the system: center point of the middle cross section and bearings at both ends. Additionally, each time window was generated with varying physical properties: rotational frequency of the rotor, bearing clearance and support stiffness. The dataset consisted of 11984 of such 9 second time windows. Displacement data from bearings was processed to acceleration numerically. An example of bearing acceleration data in time domain is demonstrated in Figure 38.Example of center-point movement in time and frequency domains is demonstrated in Figure 39.

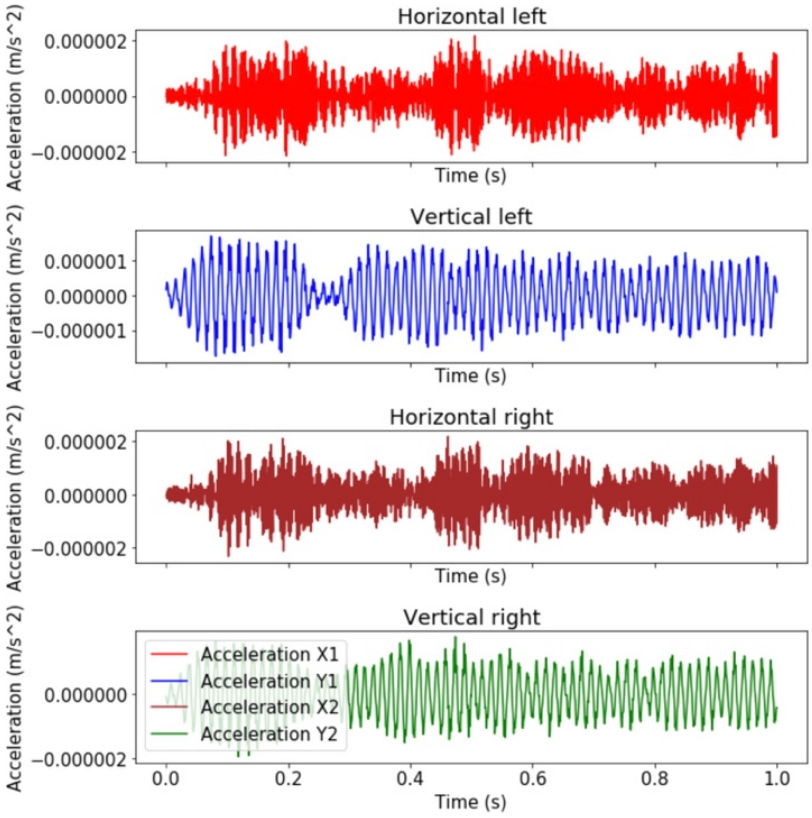


Figure 38 Subsequence of acceleration data in time domain from simulation time window with 14.5 Hz rotating speed.

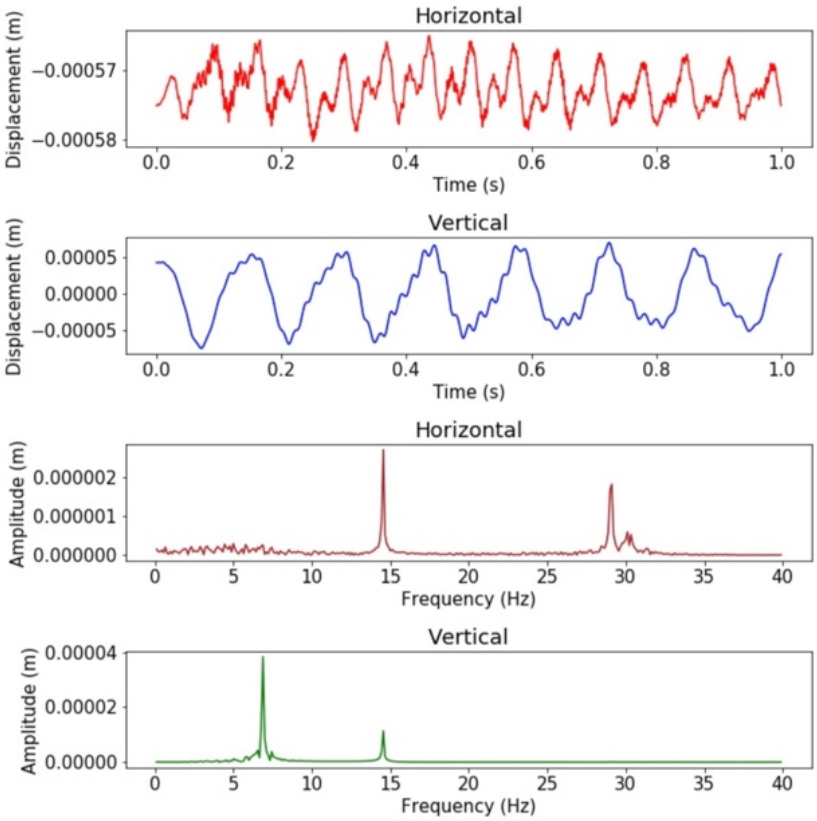


Figure 39 Subsequences of horizontal and vertical center point movement corresponding to accelerations in Figure 38.

This kind of dataset enabled simulation to observation experiment, since it included similar input and target signals to the observation dataset. As is demonstrated in Table 2,both datasets included center-point movement of the middle cross-section of the rotor as targets. Furthermore, they both included acceleration data from the bearings as features. Lastly, all data in both datasets were sampled with the rotor in constant rotating speed and after signal processing all data was in the same sampling rate of 2000 Hz.

The datasets had also some dissimilarities. The vertical target signals show different vibrational shapes between the datasets, as can be seen from examples in Figure 35 and Figure 39. The highest amplitude in the vertical vibration frequency domain is not the first harmonic component corresponding to the rotational speed. This difference might be caused by the chosen eigenmodes of the simulated rotor. Another semantic dissimilarity is that the feature signals were not from the exact same position. The observation dataset included acceleration from bearing housings whilst the simulation dataset included accelerations from bearings.

3.3 Preprocessing datasets

Overall, neural networks learn from data without complex feature engineering. However, most of the time they still tend to learn faster and with better results if some feature engineering is done. This thesis divides feature engineering in two categories: signal processing and preprocessing. Signal processing includes all necessary signal processing steps for acquiring a full dataset. For observation dataset, signal processing steps were introduced previously in Section 3.1, including the four-point method and resampling. With simulation dataset, signal processing included only computing gradients numerically from bearing displacement signals. Preprocessing steps are the necessary dataset processing steps

taken before forward propagating the sequences through a neural network. The preprocessing steps included feature scaling, time window divisions and test and training dataset splitting. Datasets used in the experiments are presented in Table 2.

Table 2 Summary of datasets used in this thesis.

Dataset	Files	Data per file	Varying physical properties between files	Sampling frequency	Signals per file
Observation data	1483	100 revolutions	-Rotating speed -Foundation stiffness	2000 Hz	1. CPM (Horizontal, Vertical), Target signal 2. Forces (2 x Horizontal, 4 x Vertical), Feature signal 3. Accelerations (2x Horizontal, 2x Vertical), Feature signal
Simulation data	11984	9 seconds	-Rotating speed -Foundation stiffness -Bearing clearance	2000 Hz	1. CPM (Horizontal, Vertical), Target signal 2. Accelerations (2x Horizontal, 2x Vertical) Feature signal

Feature scaling was done with min-max scaling. Since all the signals were somewhat similar to sine waves having both negative and positive values, the scale for all signals was chosen between -1 and 1. The signals were scaled with Equation 10 independently and along the whole sequence in a file. For example, in observation to observation experiments this means that $\boldsymbol{X}$ is a sequence with full 100 rounds of vertical center-point movement in a file. Then $\boldsymbol{X}$ was scaled with its minimum value and maximum value. The original scale can be retrieved by inverting the feature scaling equation. Inverted feature scaling is demonstrated in Equation 11.

$$\boldsymbol{X}' = 2\frac{\boldsymbol{X} - min\,(\boldsymbol{X})}{max\,(\boldsymbol{X}) - min\,(\boldsymbol{X})} - 1 \tag{10}$$

$$\boldsymbol{X} = \frac{\boldsymbol{X}' + 1}{2}(max\,(\boldsymbol{X}) - min\,(\boldsymbol{X})) + min\,(\boldsymbol{X}) \tag{11}$$

Time window division was used to control the length of sequences. This method forced all training and testing examples to have same length. The length could be for example 2000 datapoints which corresponds to 1 second of data with 2000 Hz sampling rate. It also enabled efficient use of stochastic gradient descent, since training time windows could be randomly and efficiently drawn from datasets. The division was done by extracting time windows of some length starting every N^{th} datapoint. The length of the time windows and the number

of datapoints between every time window start (stride) was determined as a part of hyperparameter optimization. In Figure 40 time windows of length 3 are extracted with a stride of 1 from full 100 round sample represented by N datapoints S_n. S_n represents one timestep with corresponding feature signal values and target signal values.

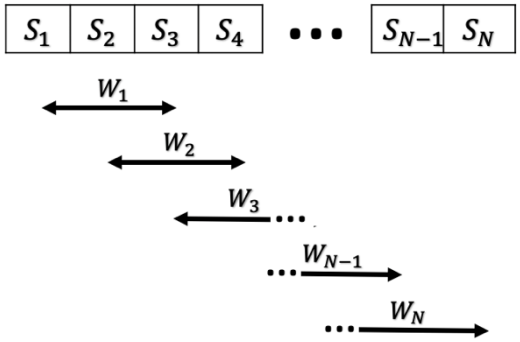


Figure 40 Time window division extracts windows w_n with some length and stride.

In O→O interpolation experiment the observation dataset in Table 2 was randomly divided so that the test files were excluded from training dataset and vice versa. This produced a test and a training dataset with unique physical parameter combinations. The interpolation division used for O→O experiment is visualized in Figure 41 where red areas represent training files and black areas represent test files. The proportions between the test and training datasets are 33% and 66% correspondingly. The stiffness control position corresponds to foundation stiffness values in the range of 4,25 and 18,32 Mn/m. The stiffness control positions closer to 0 mm correspond to higher foundation stiffnesses.

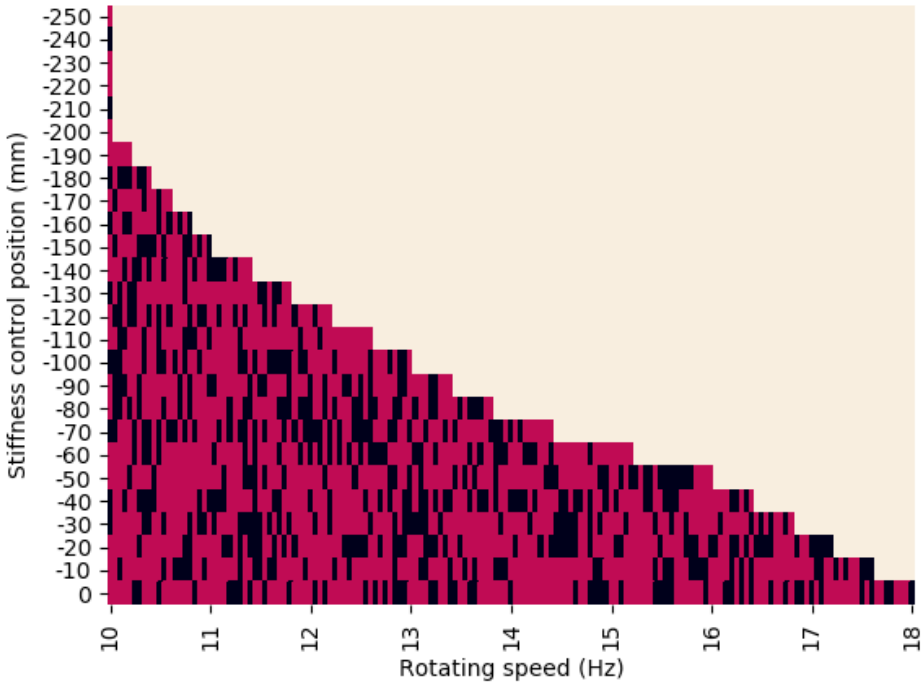


Figure 41 Division of training and testing sets in O→O interpolation experiment. The red areas are training files and the black areas testing files.

In O→O extrapolation experiment the test and training data division was similarly based on physical parameters. The train dataset included all files with rotation speed in the range of

10 to 13 Hz. The test dataset included all files with rotation speed of 13 Hz or higher. This division is demonstrated in Figure 42.

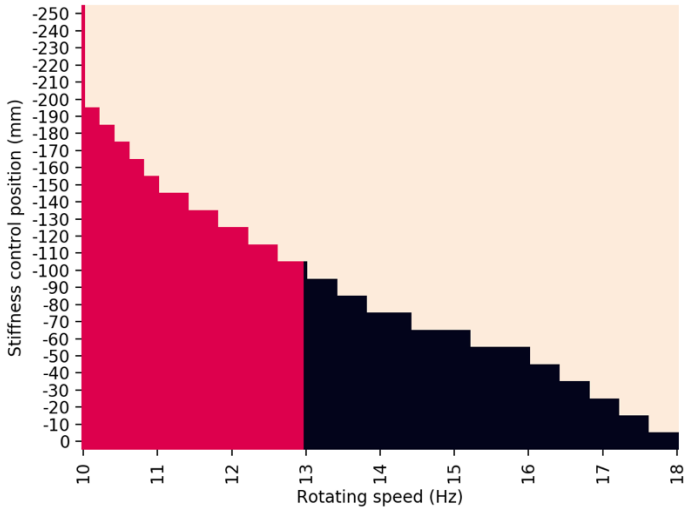


Figure 42 Division of training and testing sets in O→O extrapolation experiment. Red area includes training files and black area testing files.

In S→S and S→O experiments, the neural networks were trained with only 3600 randomly chosen 9-second files out of the available 11984 in the simulation dataset, since the amount was considered large enough. In the training set, three physical parameters were randomly varying. The parameters were: rotating frequency, bearing clearance and support stiffness. The ranges for these parameters were (4.014, 17.996), (0.01, 0.12) and (0.5, 1.0) respectively. The test set in S→S experiment was chosen similarly as in O→O interpolation, randomly and excluding the files from training set. The test set in S→O experiment consisted of every 1483 file in the observation dataset. However, the feature signals in S→O experiment were the acceleration signals, since acceleration signals were used as input signals for training the model.

3.4 Neural network model

A Long Short-Term Memory (LSTM) was chosen as the structure for the model. Based on the literature review, LSTM seems to be state of the art for time-series problems. Also, it enabled studying sequence to sequence approximations. Sequence to sequence approximation in this thesis means that for every input timestep the model approximates an output timestep. The model structure is demonstrated with a rolled graph in Figure 43 and with an unrolled graph in Figure 44.

The model had 4 recurrent layers. Each layer was simply a LSTM cell similar to the one demonstrated in Figure 15. An input at time step t was passed forward through these four cells and then scaled to the output dimension with the last linear layer. The last linear layer was similar to the output layer of the multilayer perceptron demonstrated in Figure 10. This output layer did not have any hidden layers. The output layer processed the outputs of the last LSTM cell. The output dimension of the linear layer was two, since the desired output for every timestep included a value for horizontal and vertical center-point movement. The output dimension, synonymous in literature to hidden dimension, of LSTM cells was chosen with the hyperparameter optimization, as was the number of layers in the model.

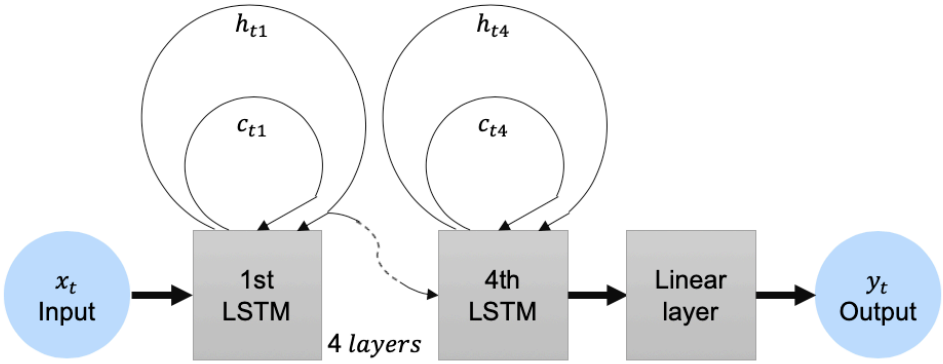


Figure 43 Rolled graph of the LSTM model.

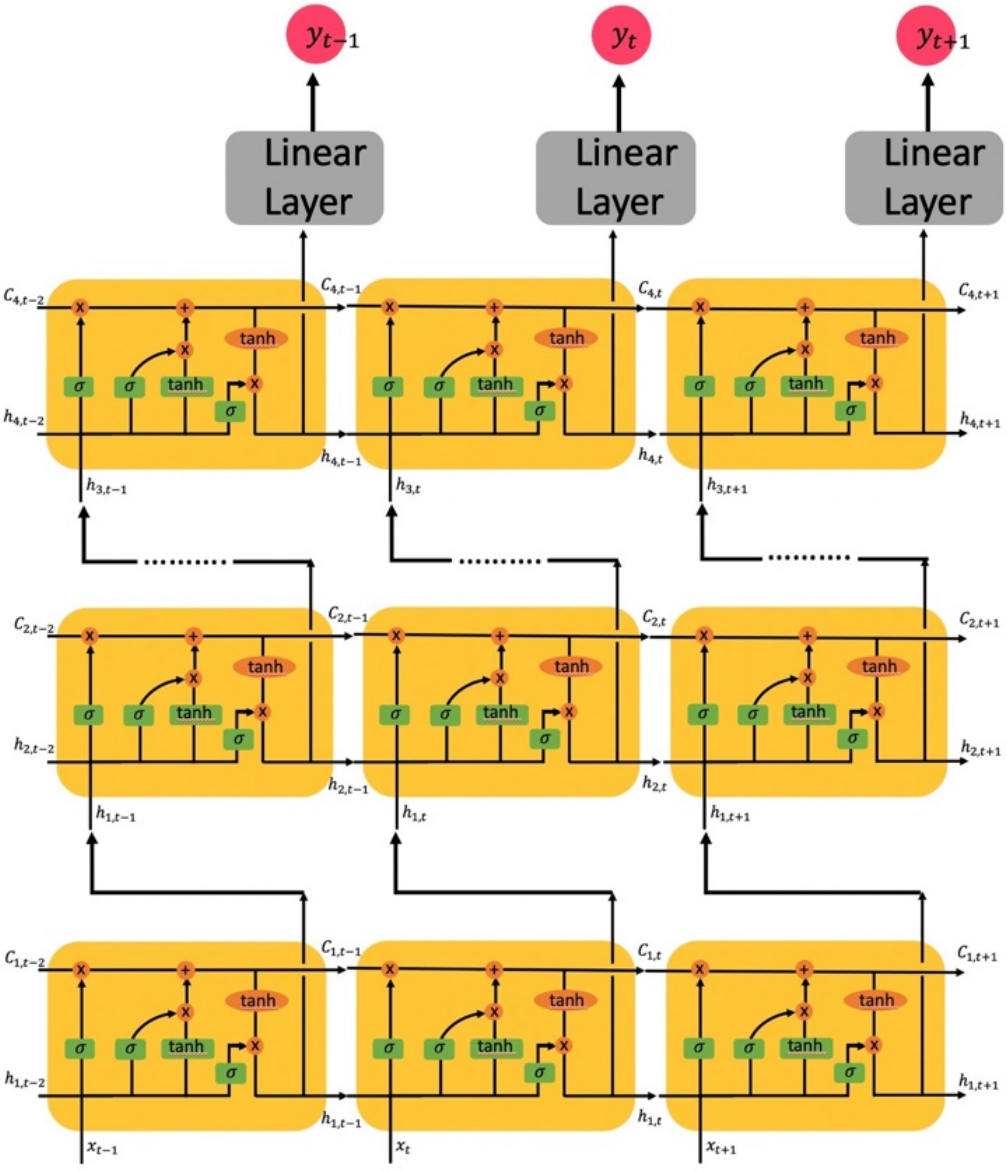


Figure 44 Unrolled graph of the LSTM model with three timesteps.

3.5 Training algorithm

The model was trained with supervised learning. For every optimization step the model was given time windows of bearing housing signals, similar to the ones in Figure 36. The model then computed the outputs for every timestep of the input signals. The output was then compared with the target center-point movement signals similar to the ones in Figure 34. With the loss function, an error term was computed by comparing the output sequence to the target sequence. The whole training algorithm is described as pseudocode in Algorithm 2.

Algorithm 2 Training algorithm steps as pseudocode.

```
best loss = inf
loss sum = 0
Do while epoch < epochs:
    shuffle order of training set
    Do while number of unprocessed subsequences < minibatch size:
        Draw a minibatch of independent feature-target signal pairs.
        Forward propagate feature signals one timestep at a time.
        loss-term = Mean-Squared-Error function (target signals, output signals)
        loss sum = loss sum + loss-term
        Backpropagate loss-term
        Take gradient descent step with Adam
        Every 500 batches:
            Average loss = loss sum / 500
            If average loss < best loss:
                Save model weights
                best loss = average loss
    epoch += 1
```

Since the model was a recurrent neural network, it was trained with backpropagation-through-time (BPTT). With BPTT, there is a risk of gradients exploding or vanishing. This risk was mitigated by making backups of the model weights every 500th forward pass that lead to smaller average loss over past 500 batches than previous best average loss was. The forward passes were computed for minibatches. A minibatch included a number of independent time windows of feature signals – target signals pairs. These minibatches were supposed to regularize the training and decrease the training time. The optimizer for the weight updates was a stochastic optimization method "Adam" introduced in Subsection 2.2.3.

3.6 Hyperparameter optimization

Hyperparameters are parameters that are not optimized during the training process. They are chosen in advance. These parameters define the structures of models, training processes and the preprocessing of datasets. The process of finding the best or nearly the best hyperparameters for a given solution is called hyperparameter optimization. In this research, some hyperparameters were optimized by random search and some were not optimized at all. Learning rate, as defined in Subsection 2.2.3 was optimized empirically after unsuccessful training trials. The hyperparameters and their optimization procedures are listed in Table 3.

Table 3 Most relevant hyperparameters of the model and their optimization methods.

Hyperparameter	Belongs to	Optimization (yes / no)	Optimization procedure	Range
Learning rate	Optimizer	yes	Empirical	0.01 … 0.001
Adam parameters (β_1, β_2)	Optimizer	no	Default PyTorch settings	N/A
# Number of Recurrent layers	Model	yes	Random search	1 … 5
Hidden dimension	Model	yes	Random search	25 … 125
Dropout	Model	yes	Random search	0.05 … 0.2
Window stride	Dataset	yes	Random search	10 … 500
Window length	Dataset	yes	Random search	500 … 4000
Batch size	Training process	yes	Random search	2 … 10
Number of Epochs	Training process	yes	Random search + Empirical	8 … 16 / 2

The random search was conducted to find a combination of hyperparameters that would provide sufficient learning capabilities for the neural network. During the optimization, 27 different combinations of values chosen randomly from ranges in Table 3 were used to train and test neural networks. The combinations were trained and tested on same 30 datasets. Same datasets were used for training and testing with the 27 combinations in order to ensure that all combinations were tested with equally difficult data. The combination that produced the best test error was chosen as the combination for the four experiments summarized in Table 1.

All 30 datasets included one file representing 100 rounds of measured data. These 30 files were chosen from the rotational speed range from 10 Hz to 16 Hz with randomly varying support stiffness. With every combination, every file was divided into smaller subsequences. The length of every subsequence depended on the window length and the number of subsequences depended on the window stride defined in Table 3. The subsequences were then divided to training and test sets corresponding to 66% and 33% of the subsequences. The test error that was monitored between the combinations was not the true test error for all 30 files. The test error was an average over 30 test errors relative to the 30 chosen files.

The goal for this optimization process was not to objectively find the best hyperparameters for the model, dataset division and training algorithm. The goal was to find parameters that likely will be suitable for training the model by overfitting the model on many different small subsets of data. Figure 45 demonstrates how train and test data was preprocessed in the hyperparameter optimization.

T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
X	0.3	0.2	0.1	0.3	0.7	0.9	0.5	0.3	0.1	0.4	0.6	0.7	0.2	0.2	0.4	0.9	0.8	0.1	0.1	0.4	0.3
Y	0.7	0.7	0.1	0.6	0.7	0.1	0.3	0.6	0.1	0.2	0.4	0.6	0.2	0.1	0.5	0.4	0.9	0.4	0.8	0.4	0.1

Figure 45 Simplification of train and test data division in the toy problem setting. X represents feature signals and Y represents target signals at timestep T.

4 Results

This chapter presents the results of the hyperparameter optimization and the four experiments in respective order. The results of the hyperparameter optimization include the parameters for the model, training algorithm and dataset preprocessing used in the four experiments. Additionally, the section for hyperparameter optimization predefines satisfying and unsatisfying approximation accuracy and demonstrates the effect the most relevant hyperparameters have on training results. The rest of this chapter is concerned of the results of the four experiments defined in Table 1. These results including O→O interpolation, O→O extrapolation, S→S and S→O experiments are shown in sections 4.2-4.5 respectively. The result sections related to the O→O experiments and S→S experiment follow a similar order. Each of these sections present the training statistics, a sample of test approximation and an overview on the whole approximation range in frequency domain. S→O result section is mostly concerned in presenting the unsatisfying results in the most meaningful manner.

4.1 Hyperparameter optimization

During the hyperparameter optimization, the performance of the models with different hyperparameter combinations was measured with Mean-Squared Error (MSE). Qualitatively, the approximations with MSE of less than 0.01 seemed to be accurate. In other words, an error less than 0.01 often meant that the center point movement approximations followed accurately the center point movement measurements. Similarly, a threshold for unsatisfactory performance could be placed at 0.1. An error over 0.1 indicated low accuracy. An example of a satisfactory approximation is demonstrated on the left and an unsatisfying approximation is demonstrated on the right in Figure 46.

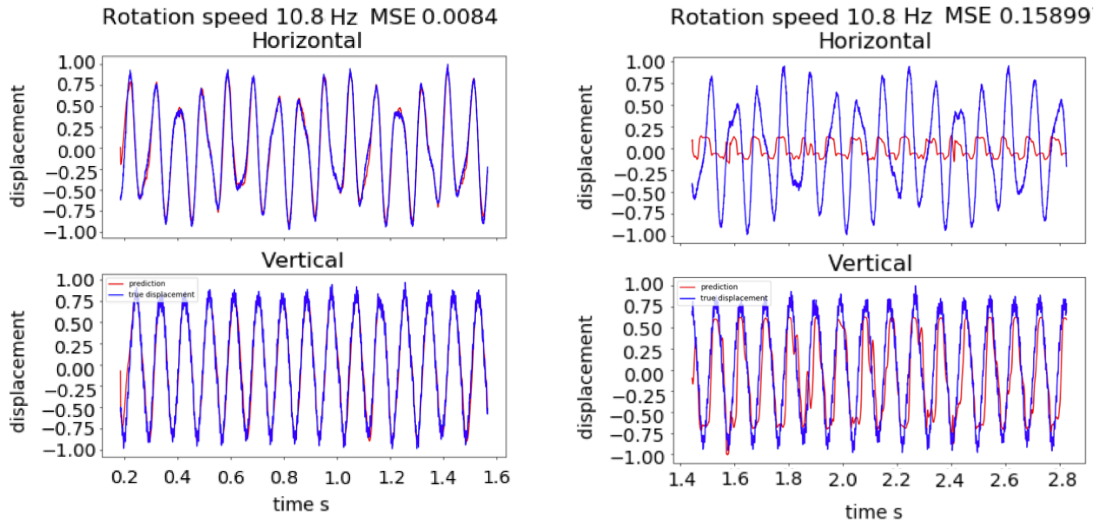


Figure 46 Time domain examples with satisfying accuracy below MSE 0.01 (left) and with unsatisfying accuracy over MSE 0.1 (right). Red curve represents approximated and blue curve represents true displacement. The displacement is scaled to the range -1 and 1.

Multiple different hyperparameter combinations seemed to produce satisfying results during the hyperparameter optimization. Figure 47 demonstrates the effect of hyperparameters on the performance of the neural network with 5 examples. From Figure 47 can be deducted that a good length for time windows was between 2000 and 3000 timesteps, a good stride between time windows was below 200 timesteps and a good hidden dimension was between 80 and 100 units. Additionally, the test accuracy grew as the number of iterations (epochs) over the training data set grew and the batch size decreased.

The combination that lead to smallest overall test error during hyperparameter optimization was chosen as the combination for the four experiments. However, the best combination was further optimized empirically, since learning rate and the number of epochs were adjusted during the four experiments. The final values and the results of the hyperparameter optimization are demonstrated in Table 4. Learning rate was empirically adjusted from 0.01 to 0.001 since the first value caused gradient vanishing occasionally. Number of epochs was reduced to 2, since the amount of available training data was large.

Table 4 Hyperparameters chosen with optimization steps.

Hyperparameter	Belongs to	Optimization procedure	Value
Learning rate	Optimizer	Random search	0.01
		Empirical	0.001
Number of recurrent layers	Model	Random search	4
Hidden dimension	Model	Random search	81
Dropout	Model	Random search	0.14
Window stride	Dataset	Random search	74
Window length	Dataset	Random search	2763
Batch size	Training process	Random search	2
Number of epochs	Training process	Random search	14
		Empirical	2

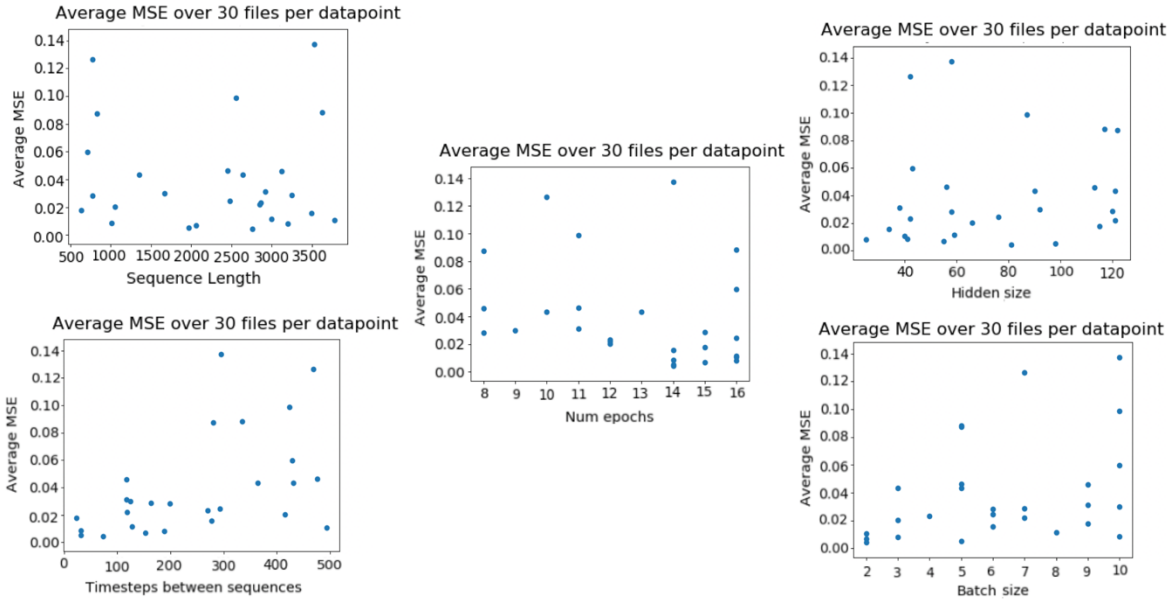


Figure 47 Average error of 27 different hyperparameter combinations over the same 30 files. Sequence length (up left) refers to window length, Timesteps between sequences (low left) refers to window stride, Num epochs (mid) refers to number of epochs, Hidden size (up right) refers to the hidden dimension and Batch size (low right) refers to minibatch dimension.

4.2 Observation to observation interpolation

In O→O interpolation experiment, the neural network training converged to an average MSE of 0.003 after around 100 000 batches had been processed. Since every batch included two time windows, the network was trained with 200 000 time windows. 200 000 time windows

corresponds to 77 hours of rotation, since one time window included 1.4 seconds of data (2763 datapoints / 2000 sampling rate). The average training error of 0.003 was computed as a running mean of the last 500 batches. Figure 48 demonstrates training converge, where one training example refers to a batch including two time windows.

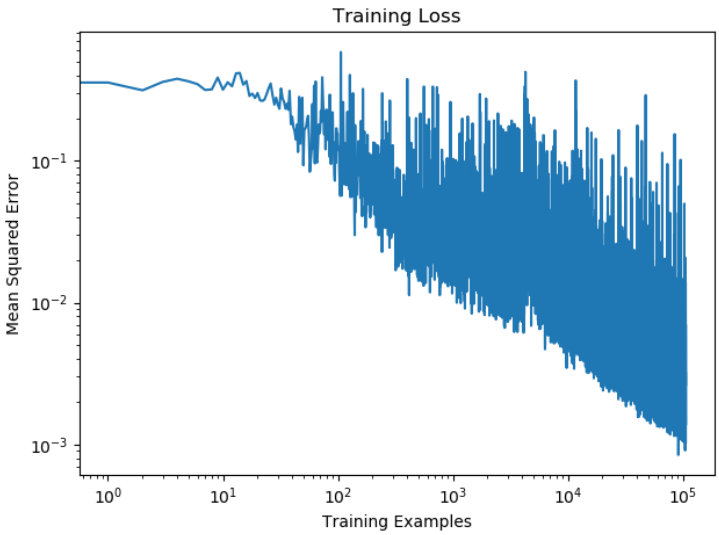


Figure 48 Training convergence in O→O interpolation experiment. One training example included two time windows of 1.4 seconds.

Test performance was measured by computing MSE for 10 randomly chosen time windows from every test file in the test dataset. The test MSE was 0.0024. Figure 49 shows the center point movement of one randomly chosen time window and the corresponding approximation in time and frequency domains. For this time window MSE was ≈ 0.0046. A typical error in the approximation can also be observed from the figure. Typically, the model approximates center-point movement inaccurately in the beginning of the time window. The error disappears after processing some input time steps, hence it is referred to as the non-repeating error.

The frequency domains of the approximated and measured center-point movements were also compared in the whole test range including varying rotating speed and foundation stiffness. The test range covers all rotating speed – foundation stiffness combinations marked with black color in Figure 41. The frequency domains were computed horizontally and vertically with FFT from one measured and one approximated 5 second time window of center-point movement from every test file in the test range. This amount of data is comparable to 41 minutes of unique rotation. The resulting measured and approximated frequency domains as a function of rotating speed are shown on the top rows in Figure 50 and Figure 51. An error surface computed as the difference between the approximation and measured frequency domains is also shown on the top right corner in Figure 50 and Figure 51. The frequency domains were also expressed in terms of harmonic components of the vibration. The lower rows in Figure 50 and Figure 51 show approximated and measured frequency domains with 1st, 2nd, 3rd, and 4th harmonic components referring to the frequency occurring once, twice, three and four times per rotation respectively.

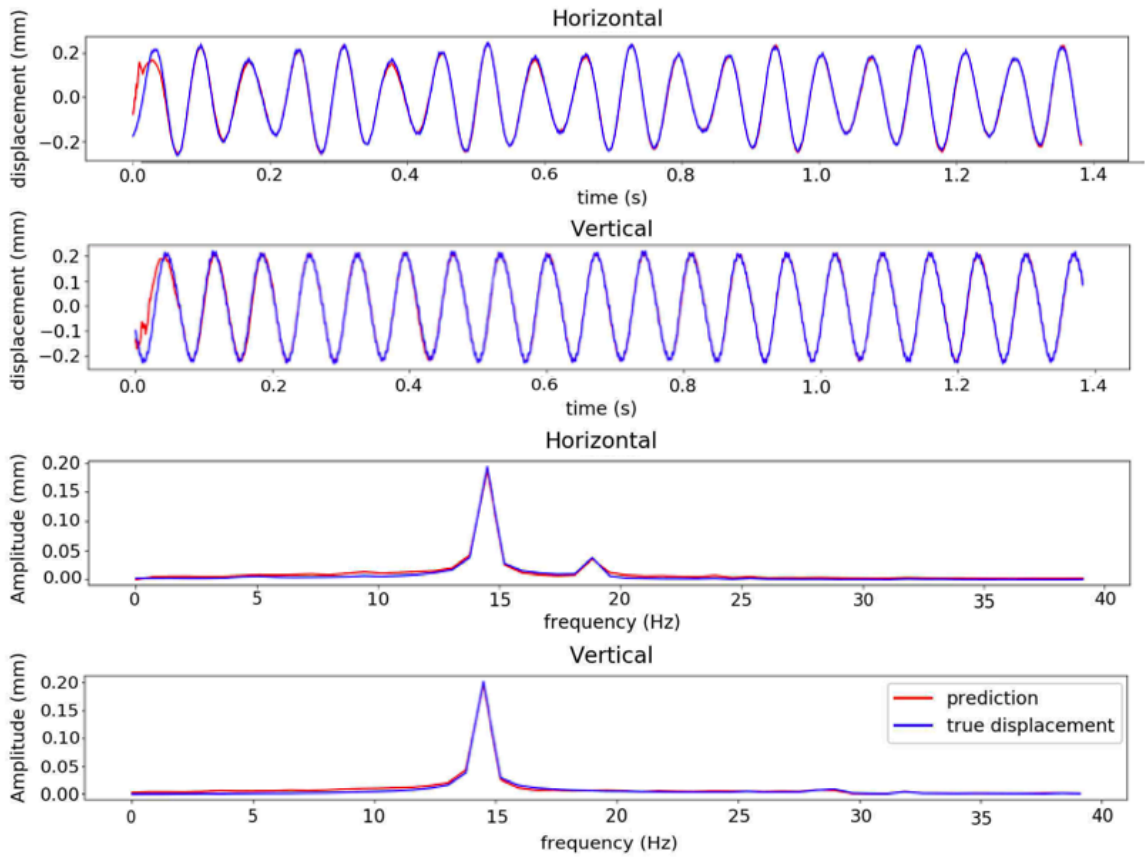


Figure 49 Time window representing center-point movement in time domain (upper two) and frequency domain (lower two). "Prediction" refers to the approximation using all 6 force sensor signals. "True displacement" is the measured center point movement.

The approximated and measured center-point movement seem to be very similar in frequency domains both vertically and horizontally. The error surface on the top right in Figure 50 and Figure 51 is barely observable in the chosen scale, since the approximation follows the measured center point movement very accurately. The most significant feature in the error surfaces is pointed by brown arrow. The arrow points to frequencies near the value 0, which are most likely caused by the non-repeating error of the approximation. The non-repeating error is shown in the beginning of the time domain plots in Figure 49.

In the upper rows of Figure 50 and Figure 51 the positions of the known natural frequencies are pointed with red arrows. The natural frequencies are known to be at locations of 30.0 Hz vertically and 21.6 Hz horizontally. The vertical natural frequency is in Figure 50 at the correct position. The horizontal natural frequency is not clearly detectable in Figure 51. The magnitude of the vibration frequency corresponding to the rotation speed horizontally and vertically is relatively high and decreasing the visibility of other frequencies.

The other frequencies are better visualized in the plots on the lower row of Figure 50 and Figure 51. All harmonics below a threshold 1.2 were manually set to zero, since the model clearly approximated the first harmonic component corresponding to the rotating speed accurately. Subharmonic resonance peaks are visible on the lower row in Figure 50. These subharmonic resonance peaks correspond to the known vertical natural frequency of 30.0 Hz. The visible subharmonic resonance peaks caused super harmonic excitations occurring twice and three times per round are pointed with red arrows. Subharmonic resonance peaks were not visible on the lower row in Figure 51, since high amplitude peaks related to time

windows with low horizontal foundation stiffness decrease the visibility of the second subharmonic resonance peak. The second subharmonic resonance peak is the only subharmonic resonance peak that theoretically should be visible in the rotating speed range, since the horizontal natural frequency is 21.6 Hz. The theoretical location of this peak is pointed with a red arrow on the lower row in Figure 51. Overall the approximated center point movement seemed to be very similar to the measured center point movement in all frequency domains.

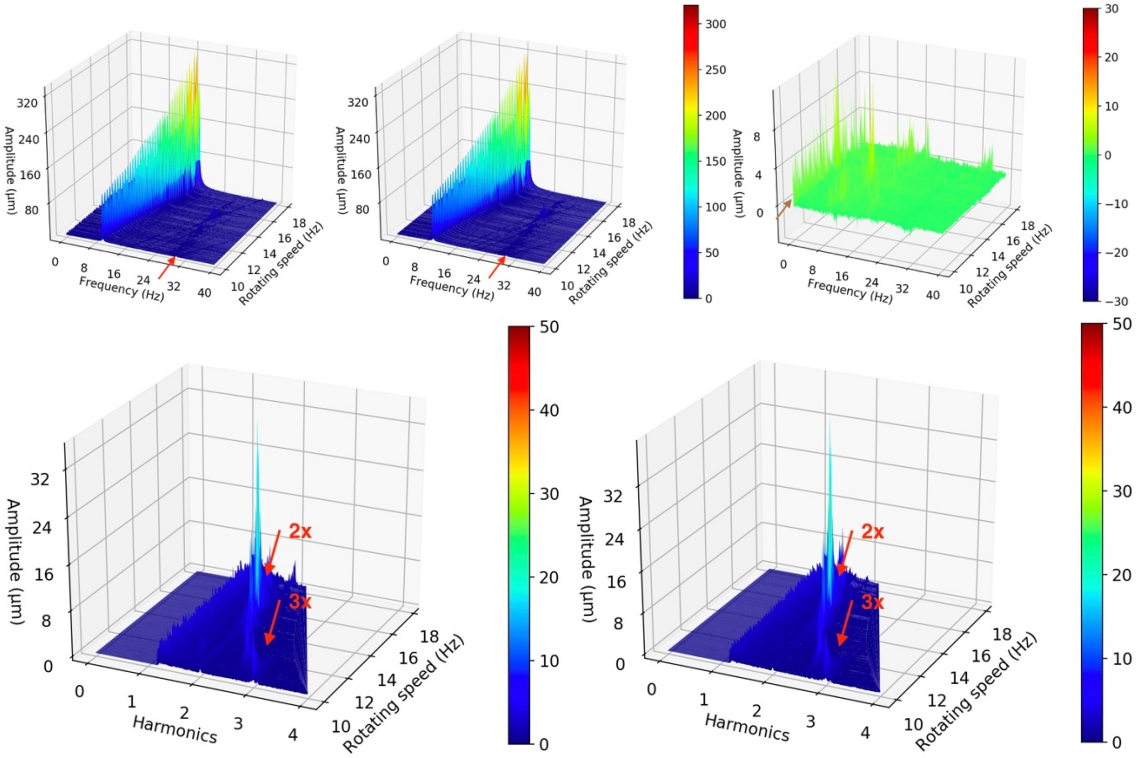


Figure 50 Vertical frequency response of the rotor as a function of rotational speed computed from the approximations (top left) and measurements (top middle). The difference between top left and top middle frequency domains (top right) is included. Vertical frequency response expressed in harmonic components computed from approximations (low left) and measurements (low right) is also included. Natural frequency is pointed by the red arrow (top left and top middle), and subharmonic resonance spikes are pointed by red arrows (low left and low right). Non-repeating error is pointed with brown arrow (top right).

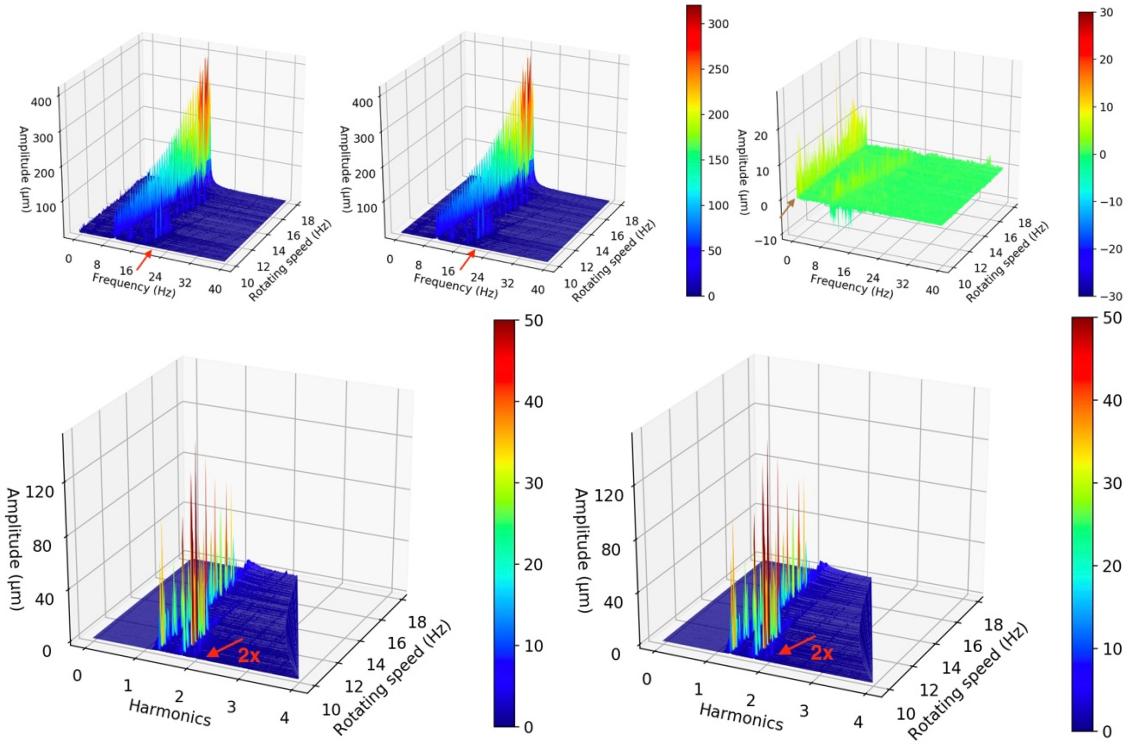


Figure 51 Horizontal frequency response of the rotor as a function of rotational speed computed from the approximations (top left) and measurements (top middle). The difference between top left and top middle frequency domains (top right) is included. Horizontal frequency response expressed in harmonic components computed from approximations (low left) and measurements (low right) is also included. Natural frequency is pointed by the red arrow (top left and top middle), and theoretical location for subharmonic resonance is also pointed by red arrow (low left and low right). Non-repeating error is pointed with brown arrow (top right).

4.3 Observation to observation extrapolation

Similar to the O→O interpolation experiment, the neural network trained in extrapolation experiment converged to an average MSE of 0.003. In the extrapolation experiment, convergence occurred with only 85 500 batches. Since every batch included two time windows, the network was trained with 171 000 time windows. 171 000 time windows corresponds to 66.5 hours of rotation, since one time window included 1.4 seconds (2763 datapoints / 2000 sampling rate). The average training error of 0.003 was computed as a running mean of the last 500 batches. Figure 52 demonstrates training convergence, where one training example refers to a batch including two time windows.

Test performance was measured by computing MSE for 10 randomly chosen time windows from every test file in the test dataset. The test dataset included all files within the rotating speed range of 13 Hz to 18 Hz, as is shown in Figure 42. The test MSE was 0.16, which was above the unsatisfactory threshold defined during the hyperparameter optimization.

Figure 53 shows the center point movement of one randomly chosen time window and the corresponding approximation in time and frequency domains. For this time window MSE was 0.15. Qualitatively, the approximation seems to be accurate in this time window. However, the error is high. The high error seems to be caused by relatively high non-repeating error and differences in the amplitudes and phase. The phase difference is small but visible, as the red approximation curve follows closely but slightly behind the blue measurement curve in time domain.

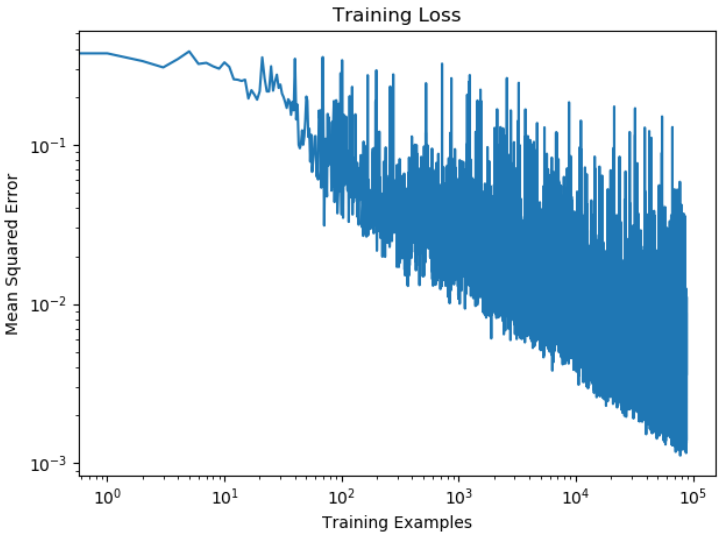


Figure 52 Training convergence in O→O extrapolation experiment. One training example included two time windows of 1.4 seconds.

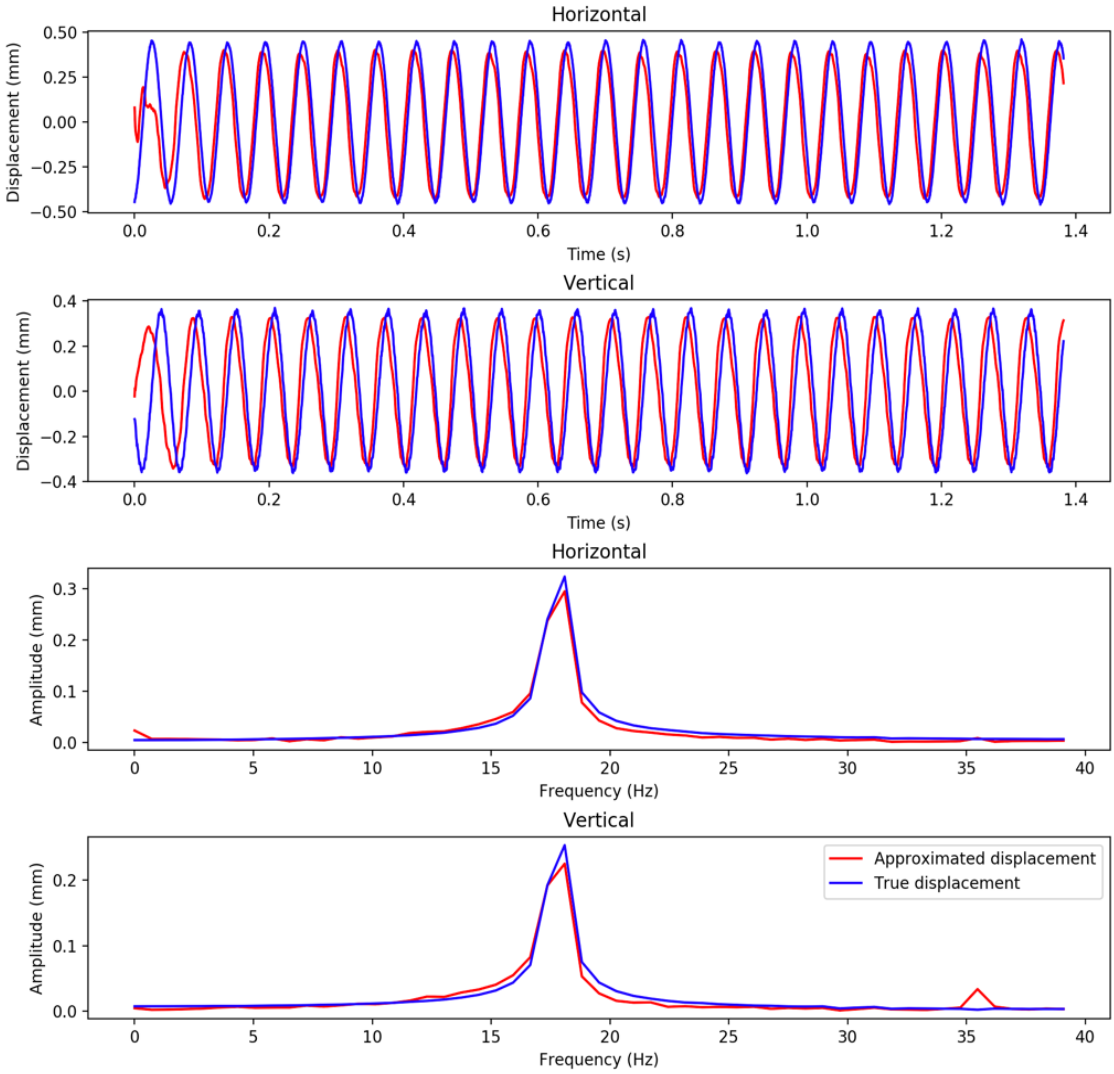


Figure 53 Time window representing center-point movement in time domain (upper two) and frequency domain (lower two). "Prediction" refers to the approximation using all 6 force sensor signals. "True displacement" is the measured center point movement.

The frequency domains of the approximated and measured center-point movements were also compared in the whole test range including varying rotating speed and foundation stiffness. The test range covers all rotating speeds and foundation stiffnesses marked with black color in Figure 42. The frequency domains were computed horizontally and vertically using FFT from one measured and one approximated 5 second time window of center-point movement from every file in the test range. This amount of data is comparable to 49 minutes of unique rotation. The resulting approximated and measured frequency domains as a function of rotating speed are shown on the top rows in Figure 54 and Figure 55. An error surface computed as the difference between the approximation and measured frequency domains is also shown on the top right corner in Figure 54 and Figure 55. The frequency domains were also expressed in terms of harmonic components of the vibration. The lower rows in Figure 54 and Figure 55 show approximated and measured frequency domains with 1st, 2nd, 3rd and 4th harmonic components referring to the frequency of a vibration occurring once, twice, three and four times per round respectively.

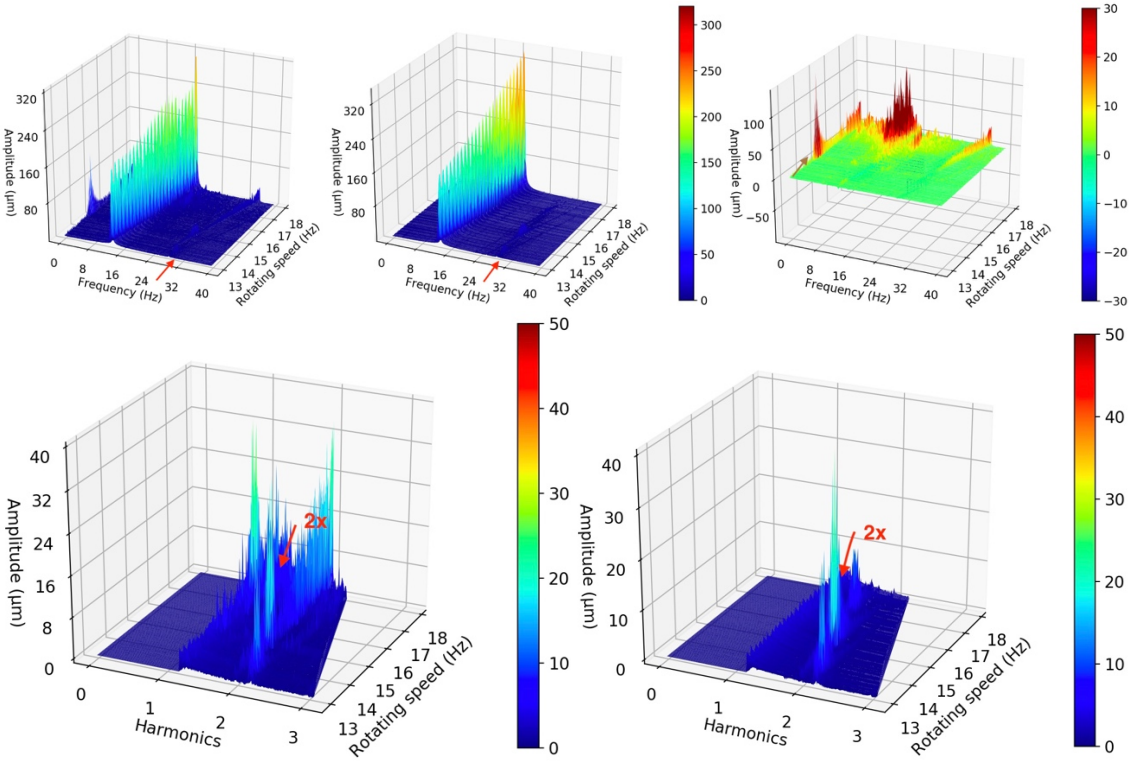


Figure 54 Vertical frequency response of the rotor as a function of rotational speed computed from the approximations (top left) and measurements (top middle). The difference between top left and top middle frequency domains (top right) is included. Vertical frequency response expressed in harmonic components computed from approximations (low left) and measurements (low right) is also included. Natural frequency is pointed by the red arrow (top left and top middle), and subharmonic resonance is also pointed by red arrow (low left and low right). Non-repeating error is pointed with brown arrow (top right).

The frequency domains of the approximated and the measured center-point movement seemed to be very similar in the rotating speed range from 13 Hz to 16 Hz both vertically and horizontally. The error surfaces on the top right in Figure 54 and Figure 55 show that the approximation error increases dramatically in the rotating speed range from 16 Hz to 18 Hz. Overall, the neural network did not learn to scale the amplitudes of the vibrations correctly over the whole test range. In addition, the non-repeating error is visibly present in

the error surface similarly as in O→O interpolation experiment. The non-repeating error is pointed by a brown arrow in the top right corner in Figure 54 and Figure 55.

In the upper rows in Figure 54 and Figure 55 the positions of the known natural frequencies are pointed by red arrows. The natural frequencies are known to be at locations of 30.0 Hz vertically and 21.6 Hz horizontally. In Figure 54, a similar subharmonic resonance peak shows at the rotating speed of 15 Hz in the frequency domains computed from both the approximated and measured center point movement. This subharmonic resonance peak is more clearly visible on the lower row in Figure 54 representing frequency domains in terms of the harmonic components, since the harmonics below 1.2 were set to zero manually in order to emphasize the frequencies with lower amplitudes. The red arrow on the lower row points to the visible subharmonic resonance peak caused by super harmonic excitation twice per rotation. In addition, approximations seem to follow closely the measured center point movement based on the lower row in Figure 55. There are no horizontal subharmonic resonance peaks in the rotating speed range for this experiment. However, the plots on the lower row in Figure 55 show that the approximated center point movement is very similar to the measured center point movement. The seemingly random amplitude peaks between the harmonics 1.2 and 2 are between the two plots.

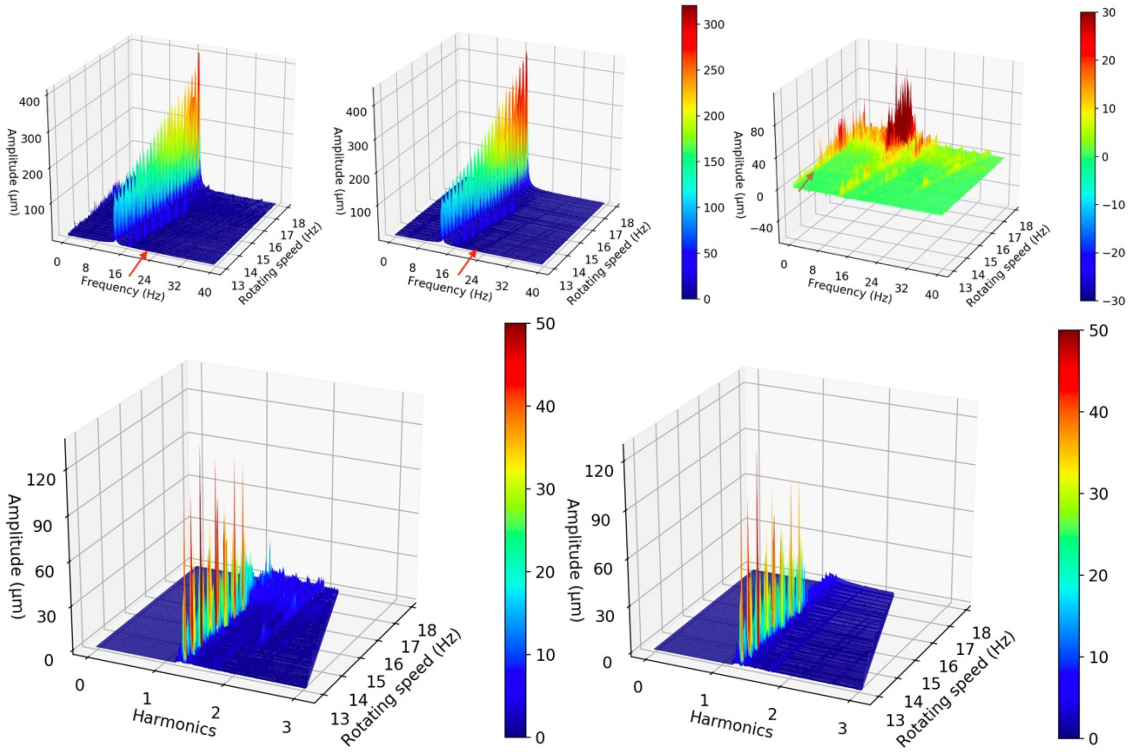


Figure 55 Horizontal frequency response of the rotor as a function of rotational speed computed from the approximations (top left) and measurements (top middle). The difference between top left and top middle frequency domains (top right) is included. Horizontal frequency response expressed in harmonic components computed from approximations (low left) and measurements (low right) is also included. Natural frequency is pointed by the red arrow (top left and top middle). Non-repeating error is pointed with brown arrow (top right).

4.4 Simulation to simulation

The training in S→S experiment converged to an average MSE of 0.012 after 221750 batches were processed. Since every batch included two time windows, the model was trained with 443 500 time windows. This amount of time windows corresponds to 172 hours of rotation, since every time window was 1.4-second-long. The average training error of

0.012 was computed as an average over the last 250 batches corresponding to 12 minutes of rotation. Figure 56 demonstrates the training convergence, where on training example refers to a batch of two time windows.

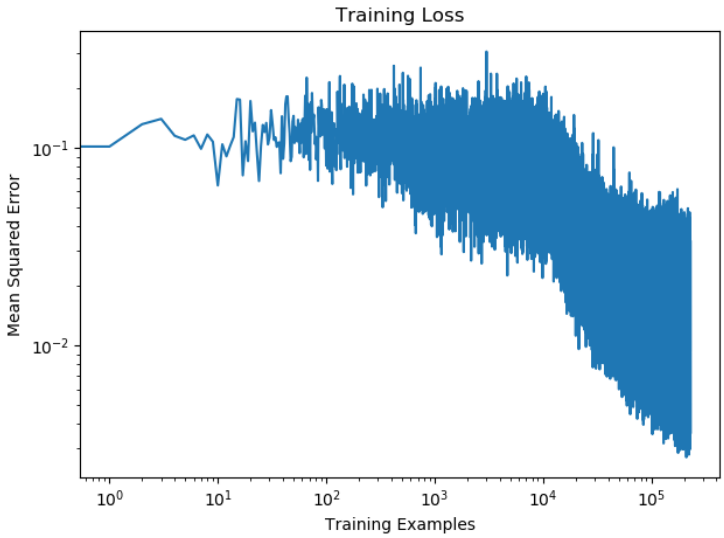


Figure 56 Training convergence in S→S experiment. One training example included two time windows of 1.4 seconds.

The test performance was measured by computing MSE for 10 randomly chosen time windows from every test file in the test dataset. The test dataset included 1200 randomly chosen test files from simulation dataset, all of which were excluded from training dataset. The test MSE was 0.12, which was above the satisfactory threshold derived during hyperparameter optimization.

Figure 57 shows the center point movement of one randomly chosen time window and the corresponding approximation in time and frequency domains. For this time window the MSE was 0.007. Similar to the O→O experiments, the approximated center-point movement demonstrates typical behavior, since at the start of the time window the model is not very accurate. This non-repeating error is visible in the upper and upper mid parts in Figure 57. Overall, the approximation seems to follow the simulated data accurately. The amplitudes are in same scale and same position in the time and frequency domains.

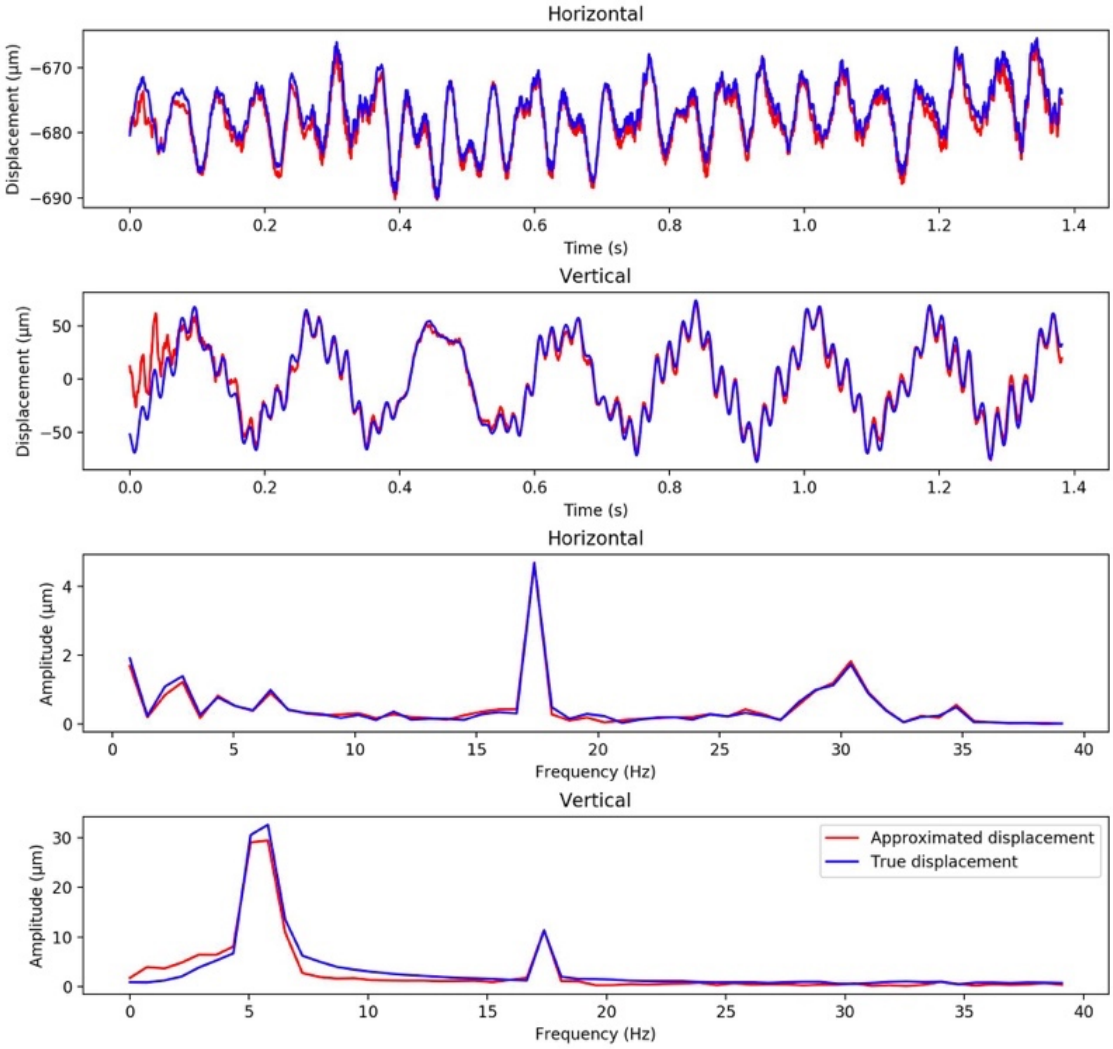


Figure 57 Center-point movement at rotating speed of 17.3 Hz, clearance 0.066 µm and foundation stiffness factor of 0.797. MSE between approximation and true displacement was 0.007.

The frequency domains of the approximated and simulated center-point movements were also compared in the whole test range including varying rotating speed, bearing clearance and foundation stiffness. The frequency domains were computed horizontally and vertically with FFT from one approximated and one simulated 1.4-second-long time window from every test file in the test dataset. This amount of data is comparable to 28 minutes of unique rotation, since the test dataset included 1200 test files.

The resulting frequency domains as a function of rotating speed are shown on the top rows of Figure 58 and Figure 59. An error surface computed as the difference between the frequency domains of approximations and simulations is also shown on the top right corner of Figure 58 and Figure 59. The frequency domains were also expressed in terms of harmonic components of the vibration. The lower rows in Figure 58 and Figure 59 show approximated and simulated frequency domains with 1st to 9th harmonic components referring to the frequency of a vibration occurring one to nine times per round respectively.

The frequency domains of the approximated and simulated center-point movements seem to be very similar in the whole test range. The shapes and amplitudes seem to be similar in both horizontal and vertical frequency domains. Horizontally the highest amplitude of around 10

µm is approximated and simulated at the same frequency of 30 Hz and rotating speed of 15 Hz. In the frequency domains of the vertical center point movement, the highest amplitudes of around 250 µm are also in the same position in the approximated and simulated plots. They appear at the first harmonic component with rotating speeds between 4 and 10 Hz. Based on the vertical frequency domains, the neural network seems to have learnt also to approximate the non-periodic vibrations that have amplitudes of around 200 µm in the rotating speeds higher than 15 Hz. However, the error surface in the top right corners of Figure 58 and Figure 59 show a relatively high error between the approximations and simulations.

Despite the relatively high error, the neural network seems to have learned to approximate center point movement from acceleration from simulation model of a rotor overall. However, based on Figure 58 and Figure 59 the simulation seems to represent a different rotor to the one used for measurements. Hence, all other qualitative analysis over the results presented in Figure 58 and Figure 59 is deemed unnecessary.

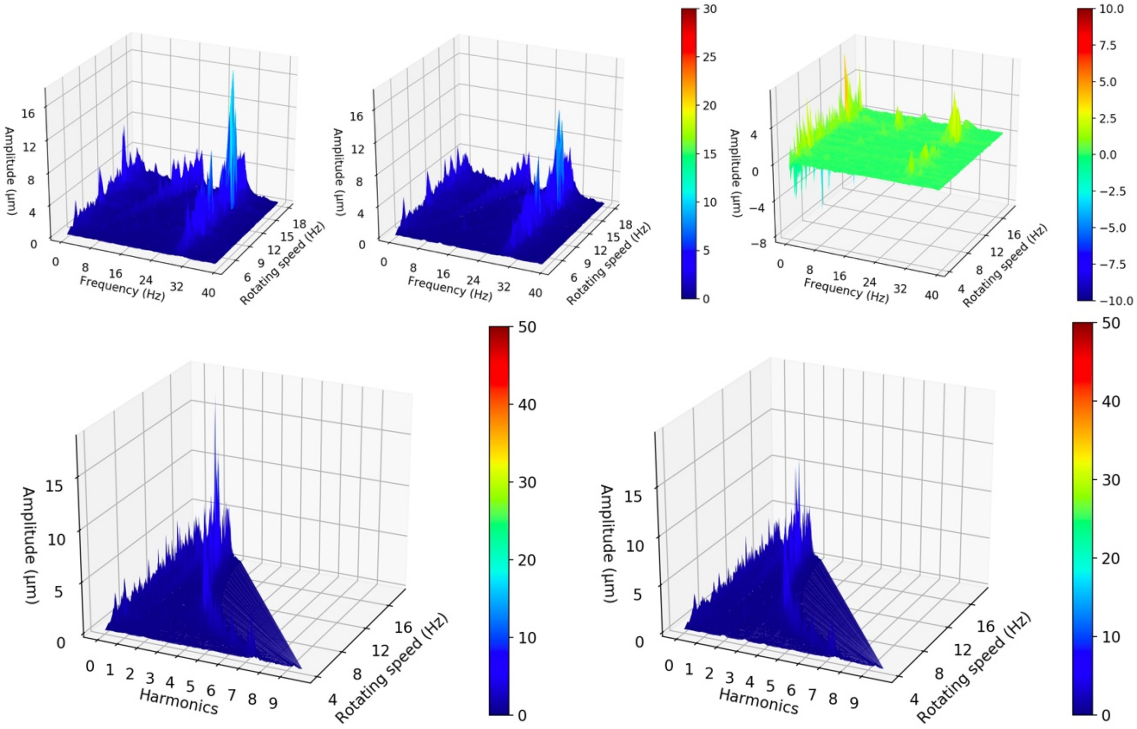


Figure 58 Horizontal frequency response of the rotor as a function of rotational speed computed from the approximations (top left) and simulations (top middle). The difference between top left and top middle frequency domains (top right) is included. Horizontal frequency response expressed in harmonic components computed from approximations (low left) and simulations (low right) is also included.

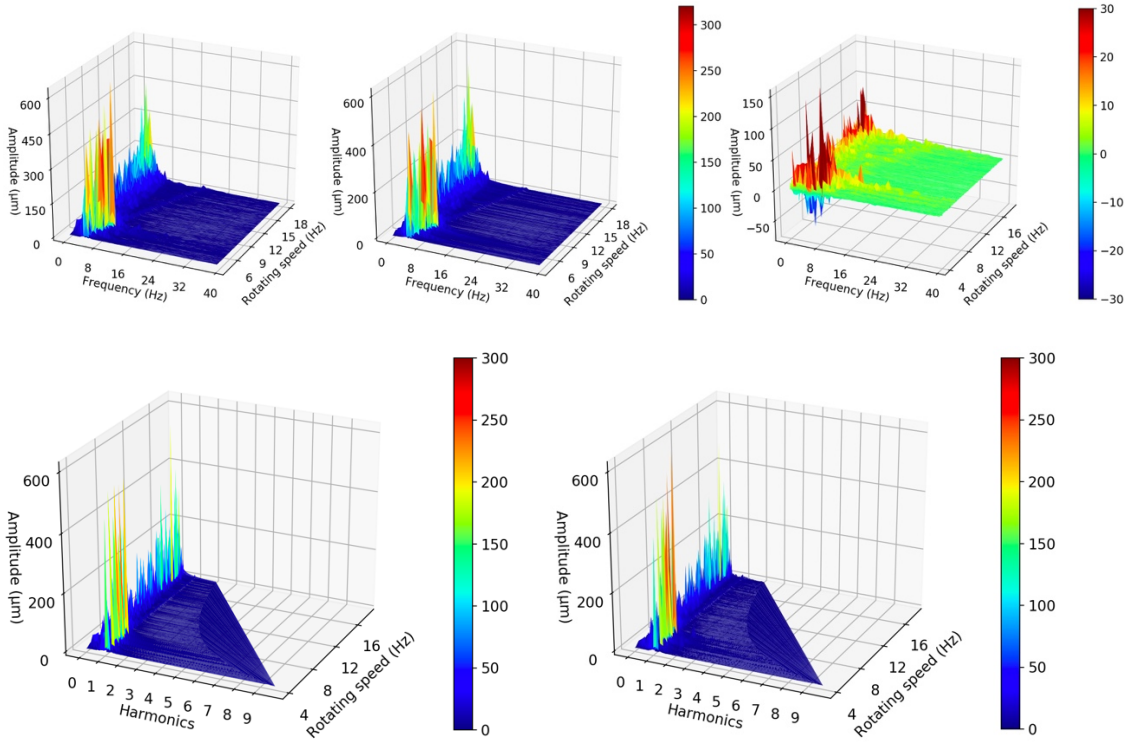


Figure 59 Vertical frequency response of the rotor as a function of rotational speed computed from the approximations (top left) and simulations (top middle). The difference between top left and top middle frequency domains (top right) is included. Vertical frequency response expressed in harmonic components computed from approximations (low left) and simulations (low right) is also included.

4.5 Simulation to observation

The same model that was trained in S→S experiment with simulation data was also tested on the measured data presented in Table 2. The feature signals in the test set were horizontal and vertical acceleration signals from both bearing housings. For the test performance, 20 time windows of 1.4 seconds from every 1483 file from the speed range of 10 to 18 Hz was randomly sampled. The test set corresponded to 11.5 hours of rotation. The test loss was then computed as an average MSE for all 11.5 hours of rotation. The average test loss was 0.48.

The average test loss indicates that a neural network trained with the simulation dataset could not approximate center-point movement of the real rotor from acceleration signals measured from the bearing housings. Additionally, the MSE distribution over the test dataset seemed to grow as a function of the rotating speed, as Figure 60 shows.

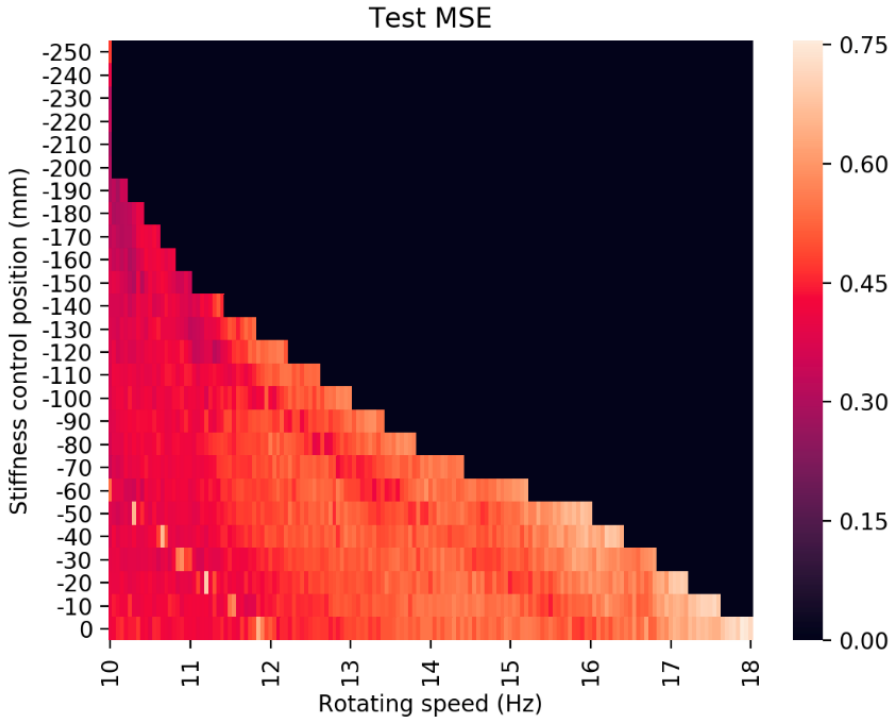


Figure 60 MSE distribution over the measured dataset of the model trained with simulated data.

However, MSE between the center-point movement signals in time domain seemed to evaluate the performance of the model poorly. Qualitatively, the performance of the model in fact seemed to grow as the rotating speed grew in contradiction to the performance evaluated with MSE. Figure 61 demonstrates approximated and measured displacements. Based on these approximations and other qualitative inspections, it seemed that the neural network approximated displacements more accurately in the higher rotating speeds than in the lower. For 16.5 and 18 Hz rotating speeds the approximated displacements in the frequency domain had the highest amplitude nearly at the right frequency. Furthermore, at rotating speed of 18 Hz the horizontal component of the center-point movement in the frequency domain had highest magnitude of around 200 µm while the measured was around 400 µm.

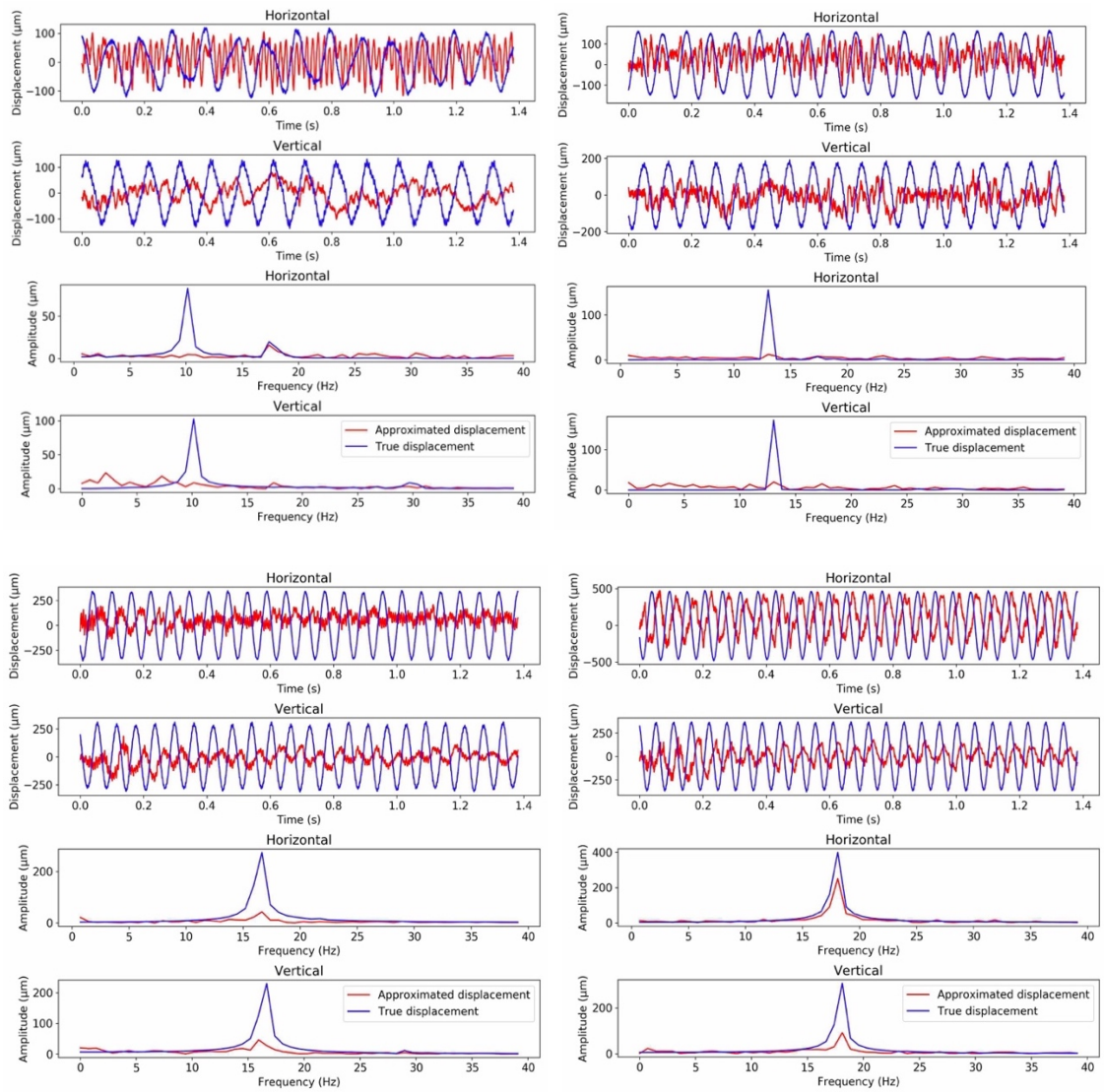


Figure 61 Approximated (red) and measured (blue) center-point movement sequences in time and frequency domain. Rotating speed, stiffness control position and MSE: 10.05 Hz, -70 mm and 0.4 (up left), 13.1 Hz, -40 mm and 0.52 (up right), 16.5 Hz, -30 mm and 0.6 (low left), 18.0 Hz, 0 mm and 0.82 (low right).

4.6 Comparison of results

As Table 5 demonstrates, training loss converged to 0.003 in both O→O experiments. The model trained in S→S experiment converged to the highest training error of 0.012 also using around 2.5 times more data in hours of rotation than the models trained for O→O experiments. Seconds of rotation before accuracy in Table 5 is a measure for the time the approximation is inaccurate at the beginning of the sequence. The model trained in O→O interpolation experiment was the fastest to reach accuracy. S→O did not result in accurate approximation.

The model trained in O→O interpolation experiment also had the smallest test error. By contrast, the model trained in O→O extrapolation had a 67 times higher test error. Moreover, the test error in S→S experiment was 50 times higher than in the O→O interpolation experiment. The test loss in S→O experiment was the highest, at 200 times higher that of O→O interpolation. Additionally, O→O interpolation experiment was the only experiment where the test loss was smaller than the training loss.

Table 5 Comparison of results.

Experiment	Training loss	Training data (hours of rotation)	Seconds of rotation before accuracy	Test loss
O→O interpolation	0.003	77	≈ 0.05	0.0024
O→O extrapolation	0.003	67	≈ 0.1	0.16
S→S interpolation	0.012	170	≈ 0.1	0.12
S→O	0.012	170	N/A	0.48

5 Discussion

The results related to hyperparameter optimization and the four experiments are discussed in the first section of this chapter. A strong emphasis is given on generalization in the discussion related to the four experiments. Thus, overfitting is a repeating theme in this chapter. Additionally, the similarities and differences of the results between the four experiments are discussed in this chapter. The discussion related to S→S and S→O experiments also emphasizes the meaningfulness of these experiments and what can be learned from them, since results were limited by certain characteristics in the simulation data. The second section briefly discusses the limitations of the developed technique. Additionally, the third section discusses the scientific impact of this thesis and compares the developed technique briefly to the literature. Finally, the possible practical impact is discussed in the last section of this chapter.

5.1 Overview

Hyperparameter optimization

The results of the hyperparameter optimization seem to have produced applicable parameters for the model, training algorithm and data preprocessing. The model was able to learn to approximate the center point movement both from measured and simulated data with the optimized hyperparameters. It still might not be the optimal structure, since the optimization was performed with small datasets, each dataset including data from one specific file with constant rotating speed and support stiffness. Instead, the randomly searched hyperparameter combinations could have been compared using larger datasets including more files.

However, the optimization procedure might have produced less optimal model for S→S and S→O experiments, since the hyperparameter optimization datasets included six measured force signals as features. By contrast, four acceleration signals were used as features in S→S and S→O experiments. This feature domain decision might be one of the causes explaining the difference in MSE between the similar S→S and O→O interpolation experiments. Overall however, the chosen hyperparameter optimization procedure seems to have been a suitable approach for finding satisfying hyperparameters based on the results overall.

Observation to observation interpolation

O→O interpolation approximations were very accurate considering the test MSE of 0.0024. However, the small test MSE raises concerns for overfitted model. Verifying fully if a model has been overfit is complex. However, the top left and top middle plots in Figure 50 and Figure 51 can be considered as proof for overfit. In these figures, the approximated and measured center-point movements in their frequency domains seem to be almost exact copies of each other. On the other hand, the error surfaces in the top right corners of these figures show that there is some difference between the approximations and measurements, which might be considered as proof against overfit.

The top left and top middle plots in Figure 50 and Figure 51 might not evaluate learning at the correct resolution, since the amplitude corresponding to the rotational frequency dominates the scale in these plots. By contrast, the lower row in Figure 50 and Figure 51 present the same frequency domains at different scale and resolution without the dominating first harmonic component. These vertical frequency domains expressed in terms of harmonic components in Figure 50 show that the model approximates the natural frequency and the related resonant behavior accurately. Figure 51 shows similar results for the horizontal direction. Despite high accuracy, at this scale the approximated vertical component does not look like the exact copy of the measured in the frequency domain. A vibration just below

the second harmonic component, probably related to some bearing excitation, visibly differs between the approximation and measurement in Figure 50.

The division between the test and training sets shown in Figure 41 demonstrates that test set including 33% of the files with unique 100-round-long samples were excluded from the training set. The test samples were measured with unique foundation stiffness and rotating speed combinations. Considering that the model was not trained on 33% of the foundation stiffness – rotating speed combinations and that the test loss was smaller than the training loss, the model might not be overfitted. However, the 66% of the whole dataset used for training might represent the interpolation range so densely that there is no significant difference between the nearest test files and training files.

Another statistical fact against overfitting is that the training converged after 1.2 epochs. Thus only 20% of the time windows in the training set was processed twice and 80% was processed once. However, multiple 1.4 second time windows could be drawn from every file including 100 rounds. These 100 rounds correspond to up to 10 seconds of unique rotation depending on the rotating frequency. The stride of 74 datapoints corresponded to 0.037 seconds of rotation between the 1.4 seconds-long time windows. Thus, from each 100-round-long sample, more than 230 of such time windows were drawn per every 10 seconds of rotation. Hence, the 80% of the unique time windows processed only once were densely overlapping.

Overall, the training data was heavily biased on the lower rotating speeds due to safety reasons in the laboratory. Figure 41 demonstrates that there was 26 times more data on rotating speed of 10.0 Hz than there was on rotating speed of 18.0 Hz. 18.0 Hz rotating speed was so close to the natural frequency of the rotor that the foundation stiffness controller could be used at only the stiffest position during data sampling. Fortunately, this bias seems to have had little effect on the results. Based on the results in Figure 50 and Figure 51, the approximations seem to be accurate also in the domain of higher rotating speeds near 18 Hz.

Observation to observation extrapolation

The model trained in the O→O extrapolation experiment converged to the same training error but faster than in the O→O interpolation experiment. Both resulted in the training error of 0.003. However, 10 hours of rotation less was needed in the O→O extrapolation training. Faster convergence might be a result of narrower training range of 10 Hz to 12.95 Hz in rotating speed. By contrast, the interpolation training range was from 10 to 18 Hz in rotating speed. Thus, the training process in this extrapolation experiment was likely less stochastic. The shorter rotating speed range is shown in Figure 42 demonstrating the training and testing data division.

The test error was 67 times higher in this extrapolation experiment than in O→O interpolation experiment. Despite the higher test error, the approximations seem to be accurate based on the frequency domains in Figure 54 and Figure 55. The approximations seem to be very accurate in the part of the test range covering rotating speeds 13 Hz to 16 Hz. However, the accuracy of the extrapolating model decreases for the rotating speeds further from this range immediately next to the training range of 10.0 Hz to 12.95 Hz. This decreasing accuracy is demonstrated in the error surfaces in the top right corners in Figure 54 and Figure 55. In the error surfaces the magnitude clearly grows in the rotating speeds of 16 Hz or higher. Overall, the approximations seem reliable over a range covered by a 3 Hz increase in the rotating speed from the training rotating speed range. Furthermore, it seems

that the model did not learn to scale the amplitudes of the vibration for the rest of the test rotating speed range (16 Hz to 18 Hz).

Despite the higher test loss and the inaccuracies with the magnitudes, the extrapolating model has learned a function between the bearing housing forces and the center point movement at some generalized level. As Figure 53 demonstrates, the approximated center point movement is quite accurate in time and frequency domains at the rotating speed of 17.9 Hz even though the loss for this time window is 0.15. All in all, the results of the extrapolating model raise confidence that neural networks can learn to approximate the lateral response of a flexible rotor in motion.

A minor problem shared by both models trained in O→O experiments is that they need to process some number of input timesteps before their approximations reach a stable accuracy. This inaccurate start has been roughly evaluated as seconds of rotation before accuracy in Table 5. The inaccuracy is also visible in the time domain approximations in Figure 49 and Figure 53. The cause for the inaccuracy is likely the lack of memories. The model reads the inputs one timestep at a time and changes the central memory state C_t at every timestep with the trained neural networks acting as gating mechanisms. It seems to take at least 0.05 seconds before the model has processed enough timesteps to recognize the signal and have the central memory state C_t setup for producing accurate outputs and short-term memory signals h_t. This problem of inaccurate starts would most likely be solved by using bi-directional LSTMs, since the LSTM did not struggle to approximate the center-point movement in the end of the signal. Bi-directional LSTMs would probably approximate the start of the sequence correctly since they read the signal in both directions in time. The backwards-in-time-processing LSTM in a bi-directional model would have processed enough timesteps to have its memory state set up when it reaches the true start of the sequence.

Simulation to simulation experiment

The meaningfulness of the whole S→S experiment can be questioned, since the vertical component of the center point movement in the simulated dataset was not similar to the vertical component measured in the laboratory. It seems that the eigenmodes for vertical vibration are different between the simulated and the real rotor. This difference can be seen by comparing the simulated and measured frequency domains in Figure 50 and Figure 59. Also, the difference in the scale of the amplitudes in the horizontal frequency domains seems to be large, if Figure 51 is compared to Figure 58. The horizontal amplitudes are over 35 times smaller in the simulated center point movement. Obviously, the rotor datasets acquired with the simulation and from the laboratory do not represent same rotor-system. This difference could be called the reality gap between simulation and real world.

However, the simulated data has some correspondence to large flexible rotors in motion. It has periodic signals in time domain, as demonstrated in Figure 57. Also, the horizontal component seems to have natural frequency around 30 Hz, which can be seen as the curve of amplitude spikes in the lower plots in Figure 58. Horizontal natural frequency of 30 Hz is close to the natural frequency of the real rotor used in this thesis. Vertically there seems to be higher amplitude at the frequency corresponding to the rotating speed, which is represented by a higher line in the upper plots in Figure 59. Thus, this experiment can be used to verify that the developed approximation technique is applicable for other datasets than the measurement dataset.

S→S experiment seems to have been a more complex problem than the O→O interpolation experiment, since 120% more data in hours of rotation was used in S→S training process. Despite more data usage, the training and test performance was less accurate than in O→O interpolation experiment. Multiple reasons for this difference might exist. For example, the rotating speed range in the S→S experiment was 150% larger than the rotating speed range in O→O interpolation experiment. O→O interpolation experiment included time windows from the rotating speed range of 10 to 18 Hz, while S→S experiment included time windows from the rotating speed range of 4 to 18 Hz.

The simulation dataset also included files that had randomly varying bearing clearance, while the laboratory dataset did not. Randomly varying bearing clearance probably introduced more complexities to the learning problem. Additionally, the center point movement in the simulation dataset was different to the laboratory dataset. This different center point movement might have been harder to learn. Another viable explanation for the difference in learning might be related to the input signals. Six force sensor signals were used in the O→O experiments while in the S→S experiment only four acceleration signals were used as inputs. These six force input signals might have been able to provide more accurate information about the rotor in motion for the model than the four acceleration input signals.

The trained model in this experiment seems to perform above the satisfying error threshold of 0.1 with the test set. However, based on frequency domains in Figure 58 and Figure 59, the model has learned to approximate the center-point movement from the acceleration signals in simulation domain with relatively high accuracy. The frequency domains are highly similar in shapes in these figures. Furthermore, based on Figure 57 and other test set approximations, the model seems to be accurate also in time-domain. On the other hand, the performance of the model is not as good as in O→O interpolation. The magnitudes of the error surfaces show relatively high inaccuracies. Lastly, in Figure 57 the model shows similar problems in perceiving the signal in the beginning as did the models in O→O experiments, which again raises the likelihood of the non-repeating error being of model-origin.

Simulation to observation experiment

The results for S→O experiment show that the reality gap between simulation and real objects affect the learning results. The reality gap in this research included the possible linearized parts and other simplifying assumptions in the simulation. The gap also included differences in the eigenmodes between the simulated and laboratory rotors. Based on the results, LSTM seems to need smaller reality gap between simulated and real rotor in order to learn to approximate the lateral response of the real rotor.

Another thing that most likely affected the test performance in the S→O experiment significantly was the noise in acceleration signals. In Figure 37, typical samples of acceleration signals show visible difference in noise levels in contrast to force signals in Figure 36. The signal-to-noise ratios are visibly at different magnitudes between the acceleration signals and force signals. This higher noise in the acceleration signals can also be one limiting factor for the performance of the model.

Surprisingly, the model seemed to have learned some useful behavior. It was able to approximate the horizontal center-point movement component almost correctly at 18 Hz rotating speed, as shown in Figure 61. The amplitude and frequency seem to be almost similar between the approximated and measured horizontal center point movement. The

model showed similar little useful behavior at the rotating speeds above 16 Hz. The difference in phase seems to be causing the higher MSE at the higher rotating speeds. The higher MSE is a result of the higher amplitudes in the wrong phase of the center point movements in higher rotating speeds. This might mean that higher MSE in Figure 60 actually indicates more accurate approximations.

5.2 Limitations

The highest inaccuracy was related to the time it took for the model to recognize the signal, if the test accuracy of the unsuccessful S→O experiment is neglected. All trained models shared this inaccuracy, as is demonstrated in Figure 49, Figure 53 and Figure 57. This problem is most likely related to the lack of input data processed and stored in LSTM memory, since the inaccuracy disappears after around 0.1 seconds of data has been processed. Using bi-directional LSTM would most likely remove this non-repeating inaccuracy. Bi-directional LSTMs could process the input sequences in two directions in time. Since there are no inaccuracies in the end with the normal LSTM, likely there wouldn't be at the beginning either with a bi-directional LSTM model.

Several limitations related to rotating speed likely exist. Since the technique was used to approximate the rotor response at a constant rotating speed, the accuracy of the approximations during angular acceleration is unknown. Furthermore, the rotating speed range used for training the neural network limits the reliable approximation range. In the O→O extrapolation experiment was demonstrated how the model trained with data in the rotating speed range from 10 Hz to 13 Hz approximated rotor vibration with lower accuracy in the rotating speed range above 16 Hz.

Simulations likely could be used for training LSTMs to approximate the dynamic behavior of large flexible rotors. In S→O experiment the training and testing data was very dissimilar and still the model showed some, although very little, useful behavior. With careful tuning of simulation, the reality gap could be narrowed enabling efficient training data acquisition for these data-based virtual sensors. However, the datasets used in this research were not capable to prove that simulations can be used to train a neural network that approximates sensor data, but to only demonstrate the reality gap between these two datasets.

5.3 Scientific impact

LSTM -model was successfully trained to approximate rotor response from vibration signals sampled near bearings. The applicability of the proposed deep regression technique was confirmed by using two different datasets. Although the simulation dataset might not accurately represent a real rotor, the test results in S→S experiment can still be considered as proof for the applicability of the developed model and training procedure.

Such deep regression technique is novel, since the LSTM has not been used previously to approximate rotor response from bearing related vibration signals in the time domain. Previous research using LSTM to process rotor vibration signals has mostly focused on classification tasks. By contrast, using LSTM to process rotor vibration in regression tasks has received less attention.

The closest comparable results to this thesis are presented in [80] and in [81]. In [80], LSTM was used to extract rotor rotating speed by measuring bearing housing acceleration and rotor displacement. However, extracting rotor rotating speed is probably relatively simple problem with measurement techniques, since the rotating speed usually corresponds to the frequency with the highest amplitude. In addition, [81] demonstrated how aircraft engine

vibration can be forecasted some seconds in the future with LSTM -model, which is a very different application. Overall, this thesis is another confirmation that LSTM is applicable for regression in the time domain, if the measured signals are accurate and the proposed preprocessing steps are used.

5.4 Practical impact

The virtual sensor developed in this thesis is another technique that can be used for rotor vibration monitoring. The technique can approximate rotor center-point movement of the middle cross-section in the time domain from bearing housing vibration. Additionally, the results show that enough information can be aggregated to the bearing vibration signals in order to approximate the rotor response with LSTM. Thus, these results indicate also that virtual sensors approximating other dynamic phenomena related to rotors can be developed.

The developed technique might have industrial relevance for example in manufacturing and mechanical industries where information of the displacement related to rotor vibration is needed. Moreover, the technique might be even more valuable for monitoring rotor vibration in machines where economic or geometrical constraints make direct surface measurements infeasible. The advantage of this technique is that only bearing vibration needs to be measured, and the sensor equipment directly measuring the rotor surface are not needed.

6 Conclusions

This research developed a virtual sensor for dynamic behavior of a large flexible rotor using a recurrent neural network. With the proposed technique the rotor lateral response was accurately approximated in the time domain. The virtual sensor technique is based on a long short-term memory neural network. The neural network was trained with data measured from a full-scale paper machine rotor and data acquired from a simulation model of a similar rotor. The proposed neural network model is called virtual sensor since it approximates the rotor center point movement of the middle cross-section from vibration signals sampled near the bearings in the time domain.

For training the long short-term memory based virtual sensor, four different experiments were designed. In the first experiment a neural network was trained and tested with randomly selected time windows with different rotating speed – foundation stiffness combinations from the measurement dataset. The test time windows were not used for training but still lay inside the training ranges of foundation stiffness and rotating speed. The approximated center point movement signals in this experiment were very accurate in time and frequency domains. The average test mean squared error for this experiment was 0.0024, which was computed between the approximated and measured center point movements in the time domain.

In the second experiment a neural network was trained with all the samples within the rotating speed range of 10.0 Hz to 12.95 Hz from the measurement dataset. The test dataset included all samples within the rotating speed range of 13 Hz to 18 Hz from the measurement dataset. This test dataset forced the trained model to extrapolate its approximations in the rotating speed domain. The test center point movement approximations were very accurate between the range 13 Hz to 16 Hz and even approximated the resonance peak correctly. The accuracy of the approximations suffered minor inaccuracy above the rotating speed range of 16 Hz. The average test mean square error for this extrapolation experiment was 0.12, which was computed between the approximated and measured center point movements in the time domain.

In the third experiment a similar neural network was trained and tested with randomly selected samples with different rotating speed – bearing clearance – foundation stiffness combinations from the simulation dataset. The training and test datasets included samples in the rotating speed range of 4 Hz to 18 Hz. The test center point movement approximations were accurate for the whole test set. The average test mean square error for this experiment was 0.12, which was computed between the approximated and measured center point movements in the time domain.

In the fourth experiment the same neural network trained with simulation dataset was used to approximate the center point movement from the acceleration signals in the measurement dataset. The test approximations showed very little useful behavior. The suboptimal test performance in this experiment is probably explained by excessive noise in measurement datasets acceleration signals and the reality gap between measurement and simulation datasets. The average test mean squared error for this experiment was 0.48, which was computed between the approximated and measured center point movements in the time domain.

This virtual sensor approximating rotor center point movement of the middle cross section in the time domain from bearing vibration signals achieved a good overall performance. The test performance in different experiments provides strong evidence that LSTM can learn to

approximate the dynamic lateral response of a large flexible rotor. These results can be considered scientifically valuable since at the time this thesis was written, only little scientific knowledge was available of signal to signal approximations in time domain with neural networks. Also, these results represent a part of new scientific paradigm where neural networks are applied to monitor or control real machines or even predict phenomena related to them.

The proposed method for center point movement approximation is a novel application. With further development it could be possible to integrate it to the monitoring system of a paper machine. This kind of additional information tool could help paper machine operators to prevent excessive vibration in the rotors. In addition, paper machine is likely not the only rotating machine where such a virtual sensor could be applied.

Based on the results in this thesis, a set of related future research topics is proposed. Firstly, the proposed virtual sensor could be further developed in to paper production conditions. In paper production conditions, the neural network would be trained with data sampled from a rotor affected by the paper web and external excitations transmitted through the foundation. Secondly, the model could be trained with data sampled from multiple different large flexible rotors in order to study the generalizability of the technique. Thirdly, a method to produce accurate acceleration data useful for training neural networks could be developed. Lastly, a way to measure and control the reality gap between simulation models and real rotors could be studied in order to effectively produce training data for various rotors used in various industries.

References

[1] Andreessen, M. 2011. Why Software Is Eating the World. The Wall Street Journal. [Web article]. [published: 20.08.2011]. [Referenced 24.4.2020]. Available: www.wsj.com/articles/SB10001424053111903480904576512250915629460.

[2] Desjardins, J. 2019. How much data is generated each day?. World Economic Forum [Web article]. [published: 17.04.2020]. [Referenced 24.4.2020]. Available: https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/.

[3] Moore, B. G. E. Cramming more components onto integrated circuits. IEEE Solid-State Circuits Society Newsletter. vol. 38, no. 8, 1965. ISSN 1098-4232

[4] Jordan, M. I. & Mitchell, T. M. 2015. Machine learning: Trends, perspectives, and prospects. Science. Vol. 349, no. 6245, pp. 255 LP – 260, Jul. 2015.

[5] Schmidhuber, J. 2014. Deep Learning in Neural Networks: An Overview. Neural Networks. Vol. 61 pp. 85-117. ISSN: 18792782. ArXiv ID: 1404.7828

[6] Google. Google Natural Language API. [Website]. [Referenced: 24.4.2020]. Available: https://cloud.google.com/natural-language

[7] Redmon, J., Divvala, S., Girshick, R. Farhadi, A. 2016. You Only Look Once: Unified, Real-Time Object Detection. Cvpr. DOI: 10.1109/CVPR.2016.91.

[8] Tesla. 2019. Tesla Autonomy Day. Youtube. [Website]. [Referenced 24.4.2020]. Available: https://www.youtube.com/watch?v=Ucp0TTmvqOE&t=8441s.

[9] Pinheiro, A. Brandao, I. & Costa, C. 2019. Vibration Analysis of Rotary Machines Using Machine Learning Techniques. European Journal of Engineering Research and Science. Vol. 4. no. 2. pp. 2–7. DOI: 10.24018/ejers.2019.4.2.1128.

[10] Jeong, H. Park, S. Woo, S. & Lee, S. 2016. Rotating Machinery Diagnostics using Deep Learning on Orbit Plot Images. Procedia Manuf., vol. 5, pp. 1107–1118.

[11] Zhang, S. Zhang, S. Wang, B. & Habetler, T. G. 2020 Deep Learning Algorithms for Bearing Fault Diagnostics - A Comprehensive Review. IEEE Access, vol. 8, pp. 29857–29881.

[12] Zekveld, M. & Hancke, G. P. 2018. Vibration condition monitoring using machine learning. Proc. IECON 2018 - 44th Annu. Conf. IEEE Ind. Electron. Soc., vol. 1, pp. 4742–4747, 2018.

[13] Kiviluoma, P. 2009. Method and device for in situ runout measurement of calender thermo rolls. Doctoral Dissertation. Aalto University.

[14] Väänänen, P. 1993. Turning of flexible rotor by high precision circularity profile measurement and active chatter compensation. Licensiate's thesis. Helsinki University of Technology. The Faculty of Mechanical Engineering. Espoo. Referenced: 24.4.2020

[15] Ozono, S. 1974. On a New Method of Roundness Measurement based on the Three Points Method. The University of Tokyo, Bunkyo-ku, Tokyo, Japan. Department of Precision Machinery Engineering.

[16] Koene, I. 2018. Development of wireless vibration monitoring equipment. Master's thesis. Aalto University. Department of Mechanical Engineering. Espoo. pp. 51.

[17] Standard SFS-ISO 21940-12:2016 Mechanical vibration. Rotor balancing. Part 12: Procedures and tolerances for rotors with flexible behaviour.

[18] Genta, G. Dynamics of Rotating Systems. Springer New York, 2007.

[19] Friswell, M., Penny, J., Garvey, S. & Lees, A. Dynamics of Rotating Machines. Cambridge University Press, 2010, pp. 1–16.

[20] Niskanen, J. 46474S Paperiteollisuuden koneet - TELAT. Oulun yliopisto, 1996.

[21] SFS-ISO 21940-11 : 2017 Mechanical vibration . Rotor balancing . Part 11 : Procedures and tolerances for rotors with rigid behaviour. 2017.

[22] Juhanko, J. 2011. Dynamic geometry of a rotating paper machine roll. Doctoral dissertation. Aalto University. ISBN 9789526043630.
[23] Widmaier, T. 2012. Optimisation of the roll geometry for production conditions. Doctoral Dissertation. Aalto University. ISBN 978-952-60-4879-6.
[24] Juhanko, J. 1999. Paperikoneen putkitelan dynaaminen käyttäytyminen. Licensiate's thesis. Helsinki University of Technology.
[25] Viitala, R. 2018. Effect of Assembled Bearing Inner Ring Geometry on Subcritical Rotor Vibration. Doctoral Dissertation. Aalto University. Helsinki. ISBN: 978-952-60-8196-0
.[26] Mechanical Engineering and Metals Industry Standardization in Finland, "SFS-ISO 21940-2:2017 Mechanical vibration. Rotor balancin. Part 2:Vocabulary." Finnish Standards Association, Helsinki, 2017.
[27] McMillan, R. 2004. Vibration Due to Unbalance. Rotating Machinery - Practical Solutions to Unbalance and Misalignment. Fairmont Press, Inc. pp. 43–54.
[28] Viitala, R,. Widmaier, T., Hemming, B., Tammi, K., & Kuosmanen, P. 2018. Uncertainty analysis of phase and amplitude of harmonic components of bearing inner ring four-point roundness measurement. *Precis. Eng.*, vol. 54, no. May, pp. 118–130.
[29] Turing, A. 1950. Computing Machinery and Intelligence. *MIND*, vol. 236, no. Quarterly Review of Psychology and Philosophy, pp. 433–460.
[30] Magnani, L., Aliseda, A., Longo, G., Sinha, C. & Thagard, P. 2013. Philosophy and Theory of Computational Intelligence. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-642-31673-9
[31] Overton, J. 2018. *Artificial Intelligence*. 1st edition. O'Reilly Media, Inc.
[32] Hölldobler, S. & Schweizer, L. 2014. Answer Set Programming and CLASP A Tutorial. *Young Sci. Int. Work. Trends Inf. Process. (YSIP). 77.*
[33] Browne, C. B. *et al.*, 2012. A Survey of Monte Carlo Tree Search Method. *IEEE Trans. Comput. Intell. AI Games*, Vol. 4, No. 1, pp. 1–43.
[34] Patterson, J and Gibson, A. 2017.*Deep Learning*, 1st Edition. Sebastopol: O'Reilly Media, Inc.
[35] Jung, A. 2018. Machine Learning : Basic Principles. arXiv:1805.05052v6.
[36] Ilin, A. 2019. CS-E4890 : Deep Learning Session 1 : Introduction. [Lecture notes]. [Referenced 24.4.2020].
[37] Goodfellow, I, Bengio, Y. & Courville, A. 2016. *Deep Learning*. MIT Press. [Referenced 24.4.2020]. Available: http://www.deeplearningbook.org.
[38] Zeiler, M. D. & Fergus, R. 2014. Visualizing and understanding convolutional networks. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833.
[39] Sutton, R. S. & Barto, A. G. 2018. *Reinforcement learning: An introduction*, 2nd ed. London: MIT press. ISBN 9780262039246.
[40] Urban, T. Neuralink and the Brain's Magical Future, Part 2: The Brain. *Wait But Why*. 2017. [Online]. Available: https://waitbutwhy.com/2017/04/neuralink.html. [Accessed: 25.9.2019].
[41] Rosenblatt, F. 1957. The perceptron A Perceiving And Recognizing Automaton. Cornell Aeronautical Laboratory, Inc. Buffalo, N.Y. Report No. 85-460-1.
[42] Richárd, N. 2018 The differences between Artificial and Biological Neural Networks. *Towards Data Science* [Online]. [Accessed: 24.4.2020]. Available: https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7
[43] Eger, S., Youssef, P. & Gurevych, I. 2016. Comparing Deep Learning Activation Functions Across NLP tasks. arXiv:1901.02671v1
[44] Maas, A. & Ng, A. 2013. Rectifier Nonlinearities Improve Neural Network Acoustic

Models. JMLR: W&CP Vol. 28.
[45] Xu, B., Huang, R. & Li, M. REVISE SATURATED ACTIVATION FUNCTIONS. arXiv:1602.05980v2
[46] Sun, S., Chen, W., Wang, L., Liu, X. & Liu, T. Y. 2016. On the depth of deep neural networks: A theoretical view *30th AAAI Conf. Artif. Intell. AAAI 2016*, pp. 2066–2072.
[47] Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals, Syst.*, vol. 2, no. 4, pp. 303–314.
[48] Ilin, A. 2019 "CS-E4890 : Deep Learning Session 4 : Convolutional networks." [Lecture notes]. [Referenced 24.4.2020].
[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.
[50] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, "Deep Residual Learning for Image Recognition arXiv:1512.03385v1," vol. 19, no. 2, pp. 107–117.
[51] Ilin, A. 2019. CS-E4890 : Deep Learning Session 5 : Recurrent neural networks. [Lecture notes]. [Referenced 24.4.2020].
[52] Schuster, M. & Paliwal, K. K. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681.
[53] Hochreiter, S. 1991. Untersuchungen zu dynamischen neuronalen Netzen. *Master's thesis, Inst. für Inform. Tech. Univ. Munchen*, pp. 1–71.
[54] Bengio, Y., Simard, P. & Frasconi, P. 2014. Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Network*, vol. 5, no. 2. p. 157.
[55] Hochreiter S. & Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780.
[56] Yu, Y., Si, X., Hu, C. & Zhang, J. 2019. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures *Neural Comput.*, vol. 31, pp. 1–36.
[57] Cho, K, van Merrienboer, B. & Gulcehre, C. 2014 Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. ArXiv ID: 1406.1078v3.
[58] Olah, C. 2016. Understanding LSTM Networks. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 02-Feb-2020].
[59] Sutskever, I., Vinyals, O. & Le, Q. V. 2014. Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, pp. 3104–3112. ArXiv ID: 1409.3215
[60] Graves, A. 2014. Generating Sequences With Recurrent Neural Networks. pp. 1–43. ArXiv ID: 1308.0850
[61] Graves, A and Jaitly, N. 2014. Towards end-to-end speech recognition with recurrent neural networks. *31st Int. Conf. Mach. Learn. ICML 2014*, vol. 5, pp. 3771–3779.
[62] Xiao, D, Huang, Y., Zhang, X., Shi, H., Liu, C. & Y. Li, 2018. Fault Diagnosis of Asynchronous Motors Based on LSTM Neural Network in *Proceedings - 2018 Prognostics and System Health Management Conference, PHM-Chongqing 2018*, 2019, no. 2017, pp. 540–545.
[63] Rumelhart, D. E., Hintont, G. E. & Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature*, vol. 323, no. 2, pp. 533–536, 1986.
[64] Graves, A. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in computational intelligence. Vol 385. ISBN: 978-3-642-24797-2.
[65] Lipton, Z. C., Berkowitz, J. & Elkan, C. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning pp. 1–38. ArXiv ID: 1506.00019

[66] Diederik, K. & Ba, J. L. 2014. ADAM: A Method for Stochastic Optimization. *AIP Conf. Proc.*, vol. 1631, pp. 58–62.
[67] Polyak, B. T. 1964. Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17.
[68] Ruder, S. 2017. An overview of gradient descent optimization algorithms. pp. 1–14. ArXiv ID: 1609.04747
[69] Keskar, N. S., Nocedal, J., Tang, P. T. P., Mudigere, D. & M. Smelyanskiy, 2019. On large-batch training for deep learning: Generalization gap and sharp minima. *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, pp. 1–16, 2019.
[70] Dietterich, T. 1995. Overfitting and Undercomputing in Machine Learning. ACM Computing Surveys, Vol. 27, No. 3. September 1995, p. 2.
[71] A. Ilin, 2019. CS-E4890 : Deep Learning Session 3 : Regularization. [Lecture notes]. [Referenced 24.4.2020].
[72] Ioffe, S. & Szegedy, C. 2019. Batch Normalization: Accelerating Deep Network Training by Reducing. 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings. ArXiv ID: 1609.04836
[73] Viitala, R. 2018. Dynamic radial bearing force measurement of flexible rotor. Master's thesis. University of Oulu. Oulu. p. 65.
[74] R. Viitala, Widmaier, T. & Kuosmanen, P. 2018. Subcritical vibrations of a large flexible rotor efficiently reduced by modifying the bearing inner ring roundness profile. *Mech. Syst. Signal Process.*, vol. 110, pp. 42–58.
[75] Viitala, R., Viitala, R., & Kuosmanen, P. Method and device to investigate the behavior of large rotors under continuously adjustable foundation stiffness. pp. 1–18. [Unpublished article currently in peer-review]. [Referenced 24.4.2020].
[76] Pirttiniemi, J. 2004. Roottorin Tasapainotusmenetelmän Kehittäminen. Helsinki University of Technology.
[77] Zhang, G. X. & Wang, R. K. 1993. Four-Point Method of Roundness and Spindle Error Measurements. *CIRP Ann. - Manuf. Technol.*, vol. 42, no. 1, pp. 593–596.
[78] Widmaier, T., Hemming, B., Juhanko, J., Kuosmanen, P., Esala, V.p., Lassila,A., Laukkanen, P & Haikio, J. 2017. Application of Monte Carlo simulation for estimation of uncertainty of four-point roundness measurements of rolls. *Precis. Eng.*, vol. 48, pp. 181–190.
[79] Oliphant, T.E. 2007. Python for Scientific Computing. *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 10–20, 2007.
[80] Rao, M., Li, Q., Wei, D. & Zuo, M. J. 2020. A deep bi-directional long short-term memory model for automatic rotating speed extraction from raw vibration signals. *Measurement*, vol. 158, p. 107719.
[81] El Said, A. E. R., El Jamiy, F., Higgins, J., Wild, B. & Desell, T. 2018. Optimizing long short-term memory recurrent neural networks using ant colony optimization to predict turbine engine vibration. *Appl. Soft Comput. J.*, vol. 73, pp. 969–991.